MASTER EN SCIENCES MATHÉMATIQUES

Détermination du nombre de classes pour des données symboliques multivaluées et modales

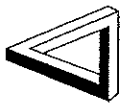
Colles, Séverine

Award date:
2003

FUNDP
Faculté des Sciences
Département de Mathématique

Rempart de la Vierge, 8
B-5000 Namur Belgique

Détermination du nombre de classes pour des données symboliques multivaluées et modales



Mémoire présenté pour l'obtention
du grade de
Licencié en Sciences Mathématiques
par

Colles Séverine

Promoteur : A. Hardy

Année Académique 2002-2003

Je tiens à remercier tout particulièrement mon promoteur, Monsieur A. Hardy, qui m'a guidée et aidée dans l'élaboration de ce mémoire. Je voudrais aussi remercier Madame P. Lallemand qui m'a permis de résoudre un certain nombre de problèmes de type informatique.

Je remercie également l'ensemble des professeurs qui ont contribué à ma formation.

Enfin, je remercie ma famille et mes amis qui m'ont soutenue tout au long de mes études.

Résumé

Dans ce mémoire, nous nous intéressons à la détermination du nombre de classes pour des données symboliques multivaluées et modales. Nous adaptons les cinq meilleures méthodes de détermination du nombre de classes issues de l'étude de Milligan et Cooper au programme de classification symbolique Sclust ainsi qu'à quatre méthodes de classification hiérarchiques (lien simple, lien complet, Ward et centroïde). Nous testons ces méthodes sur différents ensembles de données artificielles et réelles.

Abstract

In this report, we are interested in the determination of the number of clusters for multi-valued and modal symbolic data. We adapt the five best methods of determination of the number of clusters stemmed from the study of Milligan and Cooper to the program of symbolic classification Sclust as to four hierarchical methods of classification (single linkage, complete linkage, Ward and centroid). We test these methods on various artificial and real data sets.

Table des matières

Introduction **9**

1 Les données classiques **11**

1.1 Introduction 11

1.2 Les variables quantitatives 12

1.3 Les variables qualitatives 12

1.4 Vecteurs et matrice de données 14

2 Les données symboliques **16**

2.1 Introduction 16

2.2 Variables multivaluées 17

2.2.1 Définitions 17

2.2.2 Exemple 17

2.2.3 Variables multivaluées par agrégation 18

2.3 Variables de type intervalle 19

2.3.1 Définition 19

2.3.2 Exemple 19

2.3.3 Variables de type intervalle par agrégation 19

2.4 Variables modales 20

2.4.1 Définition 20

2.4.2 Exemple 20
2.4.3 Variable modale par agrégation 21
2.5 Le tableau des données symboliques 21
2.6 Données classiques vers données symboliques 23
2.7 Dissimilarités entre objets symboliques 25
2.7.1 Le cas des variables multivaluées 25
2.7.2 Le cas des variables modales 27

3 La classification automatique **28**
3.1 Introduction 28
3.2 Les structures classificatoires 29
3.2.1 Les recouvrements 29
3.2.2 Les partitions 29
3.2.3 Les hiérarchies 30
3.3 Les méthodes de classification 31
3.3.1 Les méthodes hiérarchiques 31
3.3.2 Les méthodes non-hiérarchiques 32
3.4 Les méthodes hiérarchiques agglomératives 32
3.4.1 Méthode du lien simple 32
3.4.2 Méthode du lien complet 32
3.4.3 Méthode du centroïde 33
3.4.4 Méthode de Ward 33
3.5 La méthode des hypervolumes 35
3.5.1 Processus de Poisson 35
3.5.2 Le modèle statistique 35
3.5.3 Solution statistique 36
3.5.4 Le critère des hypervolumes 36

3.6 Classification des données symboliques 37

4 La méthode des nuées dynamiques 38

4.1 Introduction 38

4.2 La notion d'inertie 39

4.2.1 Définition et propriétés 39

4.2.2 Inerties associées à une partition 39

4.2.3 Décomposition des inerties sur les classes et les variables 40

4.3 La méthode des nuées dynamiques dans le cas de données classiques 42

4.3.1 Principe général 42

4.3.2 Aspect formel 43

4.3.3 Le cas du centre de gravité 44

4.4 La méthode des nuées dynamiques dans le cas de données symboliques 46

5 Les méthodes de détermination du nombre de classes 49

5.1 Introduction 49

5.2 Les méthodes de Milligan et Cooper 49

5.2.1 La méthode de Calinski et Harabasz 49

5.2.2 La méthode de Duda et Hart 51

5.2.3 La méthode du C-index 52

5.2.4 La méthode Gamma 53

5.2.5 La méthode de Beale 54

5.3 Une méthode basée sur le critère des hypervolumes 55

6 Description du programme Sclust 57

6.1 Introduction 57

6.2 Le fichier Sodas 57
6.3 Le fichier Scluster.h 60
6.4 Le fichier calcul_scluster.cpp 61
6.4.1 La fonction Init_scluster 61
6.4.2 La fonction affec_scluster 63
6.4.3 La fonction W_scluster 66
6.4.4 La fonction Proto_scluster 69
6.4.5 La fonction main_scluster 71
6.4.6 Le fichier listing 74

7 Adaptation des méthodes de détermination du nombre de classes 79

7.1 Introduction 79
7.2 La représentation Zoom Star 80
7.2.1 2D Zoom Star 80
7.2.2 3D Zoom Star 82
7.3 Construction des hiérarchies de partitions 82
7.4 Méthodes de détermination du nombre de classes de Milligan et Cooper 84
7.5 Implémentation des méthodes et présentation des résultats dans le fichier listing 84

8 Applications 88

8.1 Introduction 88
8.2 Présentation des résultats 89
8.3 Animaux 91
8.3.1 Distance de De Carvalho 92
8.3.2 Distance L_2 94

8.3.3 Distance L_1 96
8.3.4 Conclusion 99
8.4 Boucles mérovingiennes datant du 6-8ème siècle après Jésus-Christ 100
8.4.1 Distance de De Carvalho 102
8.4.2 Distance L_2 104
8.4.3 Distance L_1 106
8.4.4 Conclusion 107
8.5 Magasins e-Fashion 109
8.5.1 Distance de De Carvalho 110
8.5.2 Distance L_2 112
8.5.3 Distance L_1 115
8.5.4 Analyse 117
8.5.5 Conclusion 117
8.6 Consommation 122
8.6.1 Distance de De Carvalho 123
8.6.2 Distance L_2 127
8.6.3 Distance L_1 132
8.6.4 Analyse 136
8.6.5 Conclusion 140
8.7 Comparaison des distances 141
8.8 Temps d'exécution 141

Conclusion **143**

A Listing du fichier classes2.cpp **145**

B Listing du fichier classes2.h **163**

C Listing du fichier classes.cpp 164

D Jeu de données merovingian.sds 168

E Jeu de données animaux.sds 171

F Jeu de données conso.sds 172

Bibliographie 177

Introduction

Aujourd'hui, il est devenu essentiel de pouvoir gérer de grandes quantités d'informations dans de nombreux domaines. Le progrès de la technologie informatique permet la collecte d'informations à grande échelle. Mais il est parfois plus primordial de ne s'intéresser qu'à des concepts sous-jacents contenus dans celles-ci. Résumer les données permet donc de mieux appréhender ces concepts et d'en extraire de nouvelles connaissances. C'est pour cela qu'il est devenu nécessaire de pouvoir analyser les données symboliques qui permettent de décrire de grands ensembles de données sans perdre trop d'informations.

Dans ce mémoire, nous nous intéresserons tout particulièrement à la classification des données. Celle-ci a de nombreux domaines d'applications comme la finance, le traitement d'images, l'industrie,... Nous étudierons plusieurs méthodes de classification qui permettent d'obtenir une partition d'une population d'objets, décrits par un ensemble de variables, en un certain nombre de classes généralement fixé a priori. Cependant, il arrive souvent de ne pas connaître le nombre optimal de classes présentes dans les données, c'est-à-dire le nombre de groupes correspondant à la partition naturelle. De nombreux indices permettant de rechercher ce nombre existent dans la littérature. C'est pour cela que Milligan et Cooper ont, en 1988, analysé une trentaine d'entre eux et les ont classés en fonction de leurs résultats. Mais ces méthodes étaient jusqu'à présent uniquement utilisées sur des données classiques et des données symboliques de type intervalle. L'objectif de ce mémoire est d'adapter ces méthodes aux données multivaluées catégoriques et modales.

Dans un premier temps, nous présenterons les données classiques et symboliques. Nous exposerons ensuite plusieurs méthodes classiques de classification et plus particulièrement la méthode des nuées dynamiques. Puis, nous nous intéresserons aux cinq meilleures méthodes de détermination du nombre de classes issues du classement de Milligan et Cooper ainsi qu'à une méthode basée sur le critère des hypervolumes. Nous détaillerons alors le programme

de classification symbolique Sclust et appliquerons pour terminer les méthodes de détermination du nombre de classes adaptées aux objets décrits par des variables multivaluées catégoriques et modales sur différents jeux de données.

Chapitre 1

Les données classiques

1.1 Introduction

Soit $\Omega = \{x_1, \ldots, x_n\}$ un ensemble de n individus. Chaque individu est caractérisé par p variables $Y_1, \ldots, Y_p$.

Une variable classique Y_j est définie par

$$\begin{array}{rcl} Y_j : \Omega & \to & \mathcal{Y}_j \\ x_k & \rightsquigarrow & Y_j(x_k) \equiv x_{kj} \end{array}$$

où
- $\mathcal{Y}_j$ est l'ensemble des valeurs prises par la variable Y_j et est appelé l'espace d'observation de Y_j ;
- x_{kj} est la valeur observée de la variable j pour l'individu x_k.

Toutes ces valeurs peuvent être rassemblées dans une matrice de données

$$\tilde{X} = (x_{kj})_{n \times p}.$$

Nous allons distinguer deux types de variables : les variables quantitatives et les variables qualitatives. Nous considérerons aussi différentes mesures de proximité δ_j entre deux individus $x, y \in \Omega$ telles que $\delta_j(x, y)$ mesure la "dissimilarité" entre ces deux éléments.

Notons que, par la suite, l'indice de la variable sera omis.

1.2 Les variables quantitatives [3]

Une variable Y est **quantitative** si l'espace d'observation $\mathcal{Y}$ est tel que $\mathcal{Y} \subseteq \mathbb{R}$.

Variable quantitative continue
Une variable quantitative Y est **continue** si elle prend un nombre infini non dénombrable de valeurs dans $\mathbb{R}$.

Dans ce cas, nous pouvons avoir :
- $\mathcal{Y} = \mathbb{R}$,
- $\mathcal{Y} = \mathbb{R}^+$, ou encore
- $\mathcal{Y} = [a, b] = \{x \in \mathbb{R} | a \leq x \leq b\}$ où $-\infty < a < b < \infty$.

La mesure de proximité $\delta(x, y)$ entre deux éléments $x, y \in \mathcal{Y}$ est fournie par la distance euclidienne, i.e.

$$\delta(x, y) = |x - y|.$$

Exemple 1.2.1
Si la variable Y représente la taille d'un individu, alors elle admet l'espace d'observation $\mathcal{Y} = \mathbb{R}^+$.

Variable quantitative discrète
Une variable quantitative Y est **discrète** si l'espace d'observation $\mathcal{Y}$ contient un nombre fini ou infini dénombrable de valeurs $\xi_i \in \mathbb{R}$.

Dans ce cas, nous pouvons avoir :
- $\mathcal{Y} = \{\xi_1, \ldots, \xi_N\} \subset \mathbb{R}$ pour un certain entier N ou
- $\mathcal{Y} = \{\xi_1, \xi_2, \ldots\} \subset \mathbb{R}$.

Exemple 1.2.2
La variable Y = "le nombre d'employés dans une entreprise" a comme espace d'observation $\mathcal{Y} = \{\xi_1, \xi_2, \ldots\} = \mathbb{N}$.

1.3 Les variables qualitatives [3]

Une variable Y est **qualitative** ou **catégorique** si le nombre de valeurs de l'espace d'observation $\mathcal{Y}$ est fini (i.e. $|\mathcal{Y}| < \infty$) et si les éléments de $\mathcal{Y}$ n'ont aucune signification numérique.

Les éléments de $\mathcal{Y}$ sont appelés dans ce cas-ci des catégories.

Variable qualitative nominale
Une variable qualitative Y est **nominale** si son espace d'observation $\mathcal{Y}$ n'a aucune structure interne.
Pour deux catégories $x, y \in \mathcal{Y}$, nous ne pouvons seulement distinguer que

$$x = y \text{ ou } x \neq y.$$

Il n'y a donc ici aucune possibilité d'ordre ou de calcul entre les catégories.

La mesure de proximité $\delta(x, y)$ entre deux catégories $x, y \in \mathcal{Y}$ est définie par :

$$\delta(x, y) = \begin{cases} 1 \text{ si } x = y \\ 0 \text{ si } x \neq y. \end{cases}$$

Exemple 1.3.1
La variable Y = "la marque d'une voiture" a pour espace d'observation $\mathcal{Y} = \{Audi, Ford, Opel, Renault\}, |\mathcal{Y}| < \infty$.

Dans certains cas, l'espace d'observation $\mathcal{Y}$ ne comprend que deux alternatives qui peuvent alors être codées par 0 et 1. Nous parlons alors de variables **binaires** ou **dichotomiques**.

Exemple 1.3.2
La variable Y = "sexe d'une personne" a pour espace d'observation $\mathcal{Y} = \{Féminin, Masculin\}$ qui peut être codé par $\mathcal{Y} = \{0, 1\}$.

Si une variable nominale a plus de deux modalités, les s catégories peuvent être codées par $0, 1, 2, \ldots, s-1$. Cependant, ces nombres ne sont que des codes et n'ont aucune signification. Aucune opération arithmétique ne peut être définie avec ces codes.

Exemple 1.3.3
L'espace d'observation correspondant à la variable Y = "marque d'une voiture" peut être codé par $\mathcal{Y} = \{0, 1, 2, 3\}$ s'il n'y a que quatre marques de voitures possibles. Les codes 0, 1, 2, 3 correspondent respectivement aux modalités "Audi", "Ford", "Opel", "Renault".

Variable qualitative ordinale
Une variable qualitative Y est **ordinale** si l'espace d'observation $\mathcal{Y}$ est muni d'un ordre total $\prec$ tel que $\forall x, y \in \mathcal{Y}, x \neq y$, nous avons soit $x \prec y$ ou soit $y \prec x$.

C'est-à-dire que les modalités prises par la variable Y peuvent être hiérarchisées entre elles mais qu'aucun calcul ne peut être défini.

Exemple 1.3.4
La variable Y = "qualité d'un produit" a pour espace d'observation $\mathcal{Y}$ = {excellent, bon, assez bon, insuffisant, très insuffisant}.

Les différentes catégories d'une variable ordinale peuvent aussi être codées de sorte que $\mathcal{Y} = \{0, \ldots, s-1\}$ où s est le nombre de modalités prises par la variable considérée. A nouveau ici, il n'est pas possible de définir des opérations arithmétiques avec ces codes.
Cependant, dans le cas d'une variable ordinale Y, les codes $0, \ldots, s-1$ peuvent souvent être considérés comme une "graduation" pour Y telle que la mesure de proximité définie par

$$\delta(x, y) = |x - y| \quad \forall x, y \in \mathcal{Y},$$

représente le nombre de catégories de $\mathcal{Y}$ strictement comprises entre x et y selon l'ordre total $\prec$ défini plus un.

1.4 Vecteurs et matrice de données [3]

Considérons un ensemble de n individus $\Omega = \{x_1, \ldots, x_n\}$ caractérisé par p variables $Y_1, \ldots, Y_p$. Notons $\mathcal{Y}_j$ l'espace d'observation de la variable Y_j, $j = 1, \ldots, p$.

En pratique, nous n'avons généralement pas besoin de connaître une seule variable mais plutôt plusieurs variables en même temps.

Nous désignerons par X le vecteur des p variables $Y_1, \ldots, Y_p$:

$$X = \begin{pmatrix} Y_1 \\ \vdots \\ Y_p \end{pmatrix} \in \chi := \bigotimes_{j=1}^{p} \mathcal{Y}_j$$

où $\chi := \bigotimes_{j=1}^{p} \mathcal{Y}_j$ est le produit cartésien des p espaces d'observation $\mathcal{Y}_1, \ldots, \mathcal{Y}_p$.

Nous noterons par $x_{kj} := Y_j(x_k)$ la valeur ou la catégorie de Y_j observée pour l'individu $x_k \in \Omega$. Pour chaque individu $x_k \in \Omega$, les p observations $x_{k1}, \ldots, x_{kp}$ sont placées dans un vecteur colonne p-dimensionnel.

$$\tilde{x}_k = X(x_k) = \begin{pmatrix} x_{k1} \\ \vdots \\ x_{kp} \end{pmatrix} \in \chi := \bigotimes_{j=1}^{p} \mathcal{Y}_j$$

Nous obtenons alors la matrice de données classiques en prenant en compte les n individus.

$$\tilde{X} = (x_{kj})_{n\times p} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} = \begin{pmatrix} \tilde{x}_1' \\ \vdots \\ \tilde{x}_n' \end{pmatrix} = (y_1, \ldots, y_p)$$

La $k^{\text{ième}}$ ligne contient les données observées pour l'individu x_k et la $j^{\text{ième}}$ colonne représente les valeurs prises par la variable Y_j. Chaque cellule contient un seul élément $x_{kj} = Y_j(x_k)$.

Exemple 1.4.1
Considérons $\Omega = \{a, b, c, d, e\}$ un ensemble de cinq magasins.
Soient :

- Y_1 = *"le type de biens vendus" (qualitative nominale) ;*
- Y_2 = *"le chiffre d'affaire annuel du magasin en euros" (quantitative continue) ;*
- Y_3 = *"la qualité des produits vendus" (qualitative ordinale) ;*
- Y_4 = *"le nombre moyen de clients par jour" (quantitative discrète).*

La matrice de données $\tilde{X}$ correspondante est de la forme suivante :

	Y_1	Y_2	Y_3	Y_4
a	alimentaire	42000	bon	528
b	vêtement	31100	excellent	354
c	électro-ménager	38000	moyen	63
d	bricolage	39500	insuffisant	145
e	parfumerie	37500	bon	24

Remarque :
Cette matrice de données est un tableau mixte car elle contient des variables qui ne sont pas toutes de la même nature.

Chapitre 2

Les données symboliques

2.1 Introduction

Les données symboliques permettent de représenter des objets plus complexes que les données classiques. Les variables définies sur ces objets peuvent prendre des valeurs comme des intervalles ou des lois de probabilité. Pour décrire les propriétés de ces objets, nous allons introduire de nouvelles variables appelées variables symboliques.

En particulier, nous allons définir trois types de variables :
- les variables multivaluées,
- les variables de type intervalle,
- les variables modales.

L'ensemble des objets E peut être soit :
- un ensemble $E = \Omega = \{x_1, \ldots, x_n\}$ d'individus appelés **objets du premier ordre** ;
- un ensemble $E = \{C_1, C_2, \ldots\}$ de classes $C_i \subseteq \Omega$ d'individus appelées **objets du second ordre**.

Des objets d'ordre supérieur peuvent être définis de manière similaire par des étapes d'agrégation successives.

2.2 Variables multivaluées [3]

2.2.1 Définitions

Une variable Y, dont l'espace d'observation est $\mathcal{Y}$, est dite **à valeurs d'ensemble** lorsque, $\forall x_k \in E$,

$$\begin{array}{rcl} Y : E & \to & \mathcal{B} \\ x_k & \rightsquigarrow & Y(x_k) \end{array}$$

où $\mathcal{B} = \mathcal{P}(\mathcal{Y}) = \{U \neq \emptyset \mid U \subseteq \mathcal{Y}\}$.

$Y(x_k)$ est donc un sous-ensemble de $\mathcal{Y}$.

Notons que nous retrouvons la définition d'une variable classique à une seule valeur dans le cas où $|Y(x_k)| = 1$, $\forall x_k \in E$, où $|Y(x_k)|$ représente le nombre de valeurs que prend la variable Y pour décrire l'individu $x_k \in E$.

Une variable à valeurs d'ensemble Y est **multivaluée** si les valeurs $Y(x_k)$ sont toutes des sous-ensembles finis de $\mathcal{Y}$. Nous avons donc que $|Y(x_k)| < \infty$, $\forall x_k \in E$.

Nous pouvons distinguer deux types de variables multivaluées.

Une variable multivaluée Y est **catégorique** si l'espace d'observation $\mathcal{Y}$ contient un nombre fini de catégories.

Une variable multivaluée Y est **quantitative** si les valeurs $Y(x_k)$ sont des ensembles finis de nombres réels : $Y(x_k) \subset \mathbb{R}$ et $|Y(x_k)| < \infty$, $\forall x_k \in E$.

2.2.2 Exemple

Considérons l'ensemble $E = \{\text{Allemagne, Belgique, France, Pays-Bas}\}$. Soient :

- $Y_1 =$ "les deux plus grandes villes du pays"
  $\mathcal{Y}_1 = \{\text{Amsterdam, Berlin, Bruxelles, Paris, ...}\}$
- $Y_2 =$ "le nombre d'habitants des deux plus grandes villes (en millions)"
  $\mathcal{Y}_2 = \mathbb{R}^+$.

Nous obtenons alors pour les quatre pays de l'ensemble E le tableau suivant :

	Y_1	Y_2
Allemagne	{Berlin,Hambourg}	{3.6,1.7}
Belgique	{Bruxelles,Anvers}	{0.9,0.4}
France	{Paris,Marseille}	{2.1,0.8}
Pays-Bas	{Amsterdam,Rotterdam}	{0.7,0.6}

La variable Y_1 est une variable multivaluée catégorique et la variable Y_2 est une variable multivaluée quantitative.

2.2.3 Variables multivaluées par agrégation

Considérons un ensemble $\Omega = \{x_1, \ldots, x_n\}$ d'individus (aussi appelés objets du premier ordre) et la variable univaluée classique $\tilde{Y}$ ayant comme espace d'observation $\mathcal{Y}$.
Soit $E = \{C_1, \ldots, C_m\}$ un ensemble de classes $C_i \subseteq \Omega$ d'individus (aussi appelées objets du second ordre).

Nous allons caractériser le comportement de ces classes en fonction de la variable $\tilde{Y}$.

A partir de la variable $\tilde{Y}$, nous allons définir une variable "globale" ou "agrégée" Y qui spécifiera les valeurs prises par la variable $\tilde{Y}$ sur les classes C_i.

Exemple 2.2.1
Soient :
- $\Omega = \{$*détenus d'une prison*$\}$;
- $E = \{C_1, \ldots, C_m\} = \{m$ *catégories de délits commis*$\}$;
- $\tilde{Y}(x_k) =$"*le nombre d'années de condamnation de l'individu* x_k" $(x_k \in \Omega)$.

La description de la catégorie C_i *est donnée par*

$$Y(C_i) = \{1, 3, 4, 5\}$$

c'est-à-dire l'ensemble des condamnations (en années) pour les détenus ayant commis le même type de délit.

2.3 Variables de type intervalle [3]

2.3.1 Définition

Une variable à valeurs d'ensemble est **de type intervalle** si $\forall x_k \in E$, l'ensemble $U \equiv Y(x_k)$ est un intervalle de $\mathbb{R}$.

Nous avons donc

$$\begin{array}{rcl} Y : \Omega & \rightarrow & \mathcal{B} \\ x_k & \rightsquigarrow & Y(x_k) = [\alpha, \beta] \end{array}$$

où α, $\beta \in \mathbb{R}$ sont tels que $\alpha \leq \beta$.
$\mathcal{B}$ est donc l'ensemble $\mathcal{I}$ des intervalles fermés bornés de $\mathbb{R}$.

2.3.2 Exemple

Considérons :
- E ={chaînes de télévisions francophones} ;
- Y ="audience du journal télévisé du soir de la chaîne pour la Belgique" (minimum et maximum sur chaque semaine du mois de mars 2003) ;
- $\mathcal{B} = \{[\alpha, \beta] \mid \alpha, \beta \in \mathbb{R}^+, 0 \leq a \leq b < \infty\}$;
- $\mathcal{Y} = \mathbb{R}$.

Nous pouvons obtenir les résultats suivants :

$$Y(x_k) = [549880, 704170]$$
$$Y(x_l) = [494500, 597360]$$

où $x_k, x_l \in E$.

2.3.3 Variables de type intervalle par agrégation

Reprenons l'exemple des détenus d'une prison. Nous pouvons décrire la catégorie C_i des détenus ayant commis un même type de délit par :

$$Y(C_i) = [\alpha, \beta]$$

où $\alpha = \min_{x_k \in C_i} \{\tilde{Y}(x_k)\}$
$\beta = \max_{x_k \in C_i} \{\tilde{Y}(x_k)\}$

Nous obtenons alors $Y(C_i) = [1, 5]$.

2.4 Variables modales [3]

2.4.1 Définition

Une variable modale Y d'espace d'observation $\mathcal{Y}$ sur $E = \Omega = \{x_1, \ldots, x_n\}$ est une variable multivaluée pour laquelle
- $\forall x_k \in E, Y(x_k) \subset \mathcal{Y}$;
- pour toute catégorie $y \in Y(x_k)$, nous associons un poids, une probabilité ou une fréquence $w(y)$ qui indique la pertinence de la catégorie y pour l'objet x_k.

Une variable **modale** Y sur un ensemble $E = \Omega = \{x_1, \ldots, x_n\}$ d'objets à valeurs dans $\mathcal{Y}$ est définie par :

$$Y(x_k) = (U(x_k), \pi_k), \forall x_k \in E$$

où
- π_k est une mesure ou une distribution (poids, probabilité ou fréquence) sur les valeurs possibles de $\mathcal{Y}$, et
- $U(x_k) \subseteq \mathcal{Y}$ est le support de π_k dans le domaine $\mathcal{Y}$.

2.4.2 Exemple

Considérons :
- $E = \{a, b, c, d\}$ un ensemble de quatre campings ;
- Y ="le pays de provenance des vacanciers d'un camping", c'est-à-dire un sous-ensemble de $\mathcal{Y}$ ={Allemagne, Belgique, France, Pays-Bas,...} comprenant différents pays ainsi que leur pourcentage de présence dans chaque camping. Donc, $\mathcal{B}$ est l'ensemble des distributions de fréquence sur $\mathcal{Y}$.

Nous obtenons alors par exemple pour le camping a :
$Y(a) =$ {(Allemagne,0.15), (Belgique,0.35), (Pays-Bas,0.40), (Angleterre,0.05), (France,0.05)}.

2.4.3 Variable modale par agrégation

Reprenons l'exemple des détenus. Considérons C_1 := la classe des dix détenus ayant commis un vol. Alors, la variable $\tilde{Y}$ qui compte le nombre d'années de condamnation vaut pour cette classe :

$$\tilde{Y}(C_1) = \{0.5, 1, 2, 3, 1, 2.5, 1.5, 2, 1, 4\}.$$

La variable modale Y qui décrit le nombre d'années de prison des détenus de la classe C_1 peut avoir la réalisation suivante :

$$Y(C_1) = \left\{ \left( ]0,1], \frac{4}{10} \right), \left( ]1,2], \frac{3}{10} \right), \left( ]2,3], \frac{2}{10} \right), \left( ]3,4], \frac{1}{10} \right) \right\}.$$

La variable Y peut être vue ici comme une variable "histogramme".

2.5 Le tableau des données symboliques [3]

Soit $\Omega = \{x_1, \ldots, x_n\}$ un ensemble de n individus. Considérons un ensemble d'objets $E = \{x_1, \ldots, x_N\}$. E peut être :

- l'ensemble des n individus, i.e. $E = \Omega = \{x_1, \ldots, x_n\}$ $(N = n)$;
- un sous-ensemble $E \subset \Omega$, i.e. un échantillon de Ω $(N < n)$;
- un ensemble $E = \{C_1, \ldots, C_m\}$ de classes $C_1, \ldots, C_m \subseteq \Omega$ d'individus $x_k \in \Omega$ $(N = m)$. Dans ce cas, E est un ensemble d'objets du second ordre.

Considérons aussi p variables symboliques $Y_1, \ldots, Y_p$, où Y_j a pour espace d'observation $\mathcal{Y}_j$, ainsi qu'un objet $x_k \in E$.

Notons par $X(x_k) \equiv (Y_1(x_k), \ldots, Y_p(x_k))'$ le vecteur des variables symboliques déterminées pour $x_k \in E$.

Ainsi, chaque objet $x_k \in E$ peut être décrit par un vecteur de données symboliques :

$$\tilde{x}_k = X(x_k) = \begin{pmatrix} x_{k1} \\ \vdots \\ x_{kp} \end{pmatrix} \in \chi = \bigotimes_{j=1}^{p} \mathcal{B}_j$$

où

- $x_{kj} = Y_j(x_k) \in \mathcal{B}_j$ est la valeur de la $j^{\text{ième}}$ variable symbolique Y_j pour l'individu x_k $(j = 1, \ldots, p)$, et
- $\bigotimes_{j=1}^{p} \mathcal{B}_j$ est le produit cartésien des p espaces d'observation $\mathcal{B}_1, \ldots, \mathcal{B}_p$.

Toutes ces données peuvent être compilées dans une matrice de données symboliques

$$\underline{X} := \begin{pmatrix} \tilde{x}_1' \\ \vdots \\ \tilde{x}_N' \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{Np} \end{pmatrix} = (x_{ij})_{N \times p}$$

où la cellule x_{ij} peut contenir soit un ensemble, soit un intervalle ou encore un histogramme.

La $k^{\text{ième}}$ ligne de ce tableau est la description symbolique de l'élément $x_k \in E$.

Exemple 2.5.1
Considérons $E = \{a_1, a_2, a_3, a_4\}$ un ensemble de quatre magazines hebdomadaires ($N = 4$).
Soient :

- *Y_1 = "le nombre d'exemplaires vendus" (minimum et maximum sur chaque semaine de l'année 2002, en milliers).*
  D'où $\mathcal{B}_1 = \mathcal{I} = \{[\alpha, \beta] = [\min, \max]$ tels que $0 \leq \alpha \leq \beta < \infty\}$.
  Y_1 est une variable de type intervalle.
- *Y_2 = "le spectre des catégories d'âge qui lisent le magazine", c'est-à-dire un sous-ensemble de $\mathcal{Y}_2 = \{A, B, C, D\}$ où A =les moins de 20 ans, B=les 20-40 ans, C =les 40-60 ans et D =les 60 ans et plus, ainsi que le pourcentage de personnes pour chaque catégorie d'âge qui lisent le magazine.*
  Donc, $\mathcal{B}_2$ est l'ensemble des distributions de fréquence sur $\mathcal{Y}_2$.
  Y_2 est une variable modale (ou variable histogramme).
- *Y_3 = "les deux principales rubriques du magazine". C'est un sous-ensemble de toutes les rubriques du magazine.*
  $\mathcal{Y}_3$ ={sport, programme tv, actualité, annonces, courrier des lecteurs,...}.
  $\mathcal{B}_3 = \mathcal{P}(\mathcal{Y}_3)$.
  Y_3 est une variable multivaluée catégorique.

La matrice de données $\underline{X}$ correspondante est de la forme suivante :

	Y_1	Y_2	Y_3
a_1	[50000,65000]	(A 0.1 ; B 0.1 ; C 0.5 ; D 0.3)	{actualité, sport}
a_2	[45000,52000]	(A 0.3 ; B 0.4 ; C 0.2 ; D 0.1)	{cinéma, programme tv}
a_3	[21000,26000]	(A 0.2 ; B 0.5 ; C 0.2 ; D 0.1)	{actualité, offres d'emploi}
a_4	[12000,14000]	(A 0.3 ; B 0.3 ; C 0.2 ; D 0.2)	{sport, annonces}

La deuxième ligne de ce tableau
$\tilde{x}_2' = ([45000, 52000],\ (A\ 0.3 ; B\ 0.4 ; C\ 0.2 ; D\ 0.1),\ \{\text{cinéma, programme tv}\})$
correspond à la description du magazine a_2.

2.6 Données classiques vers données symboliques [3]

Considérons un ensemble Ω d'individus x_k décrits par p variables classiques $\tilde{Y}_1, \ldots, \tilde{Y}_p$ d'espaces d'observation $\mathcal{Y}_1, \ldots, \mathcal{Y}_p$. Considérons aussi $E = \{C_1, \ldots, C_m\}$ une collection de classes $C_1, \ldots, C_m \subseteq \Omega$ d'individus. Ces classes sont décrites par p variables symboliques $Y_1, \ldots, Y_p$ définies de telle façon que $Y_j(C_i)$ caractérise l'ensemble $\{\tilde{Y}_j(x_k) \mid x_k \in C_i\} \subseteq \mathcal{Y}_j$ des valeurs observées pour $\tilde{Y}_j$ à l'intérieur de la classes C_i.

Exemple 2.6.1
Considérons la matrice de données classiques $\tilde{X} = (x_{kj})$ contenant $n = 15$ individus et $p = 4$ variables univaluées classiques :

- $\tilde{Y}_1$ ="*type de bien immobilier*";
- $\tilde{Y}_2$ ="*surface habitable du bien immobilier (en m^2)*";
- $\tilde{Y}_3$ ="*prix du bien immobilier (en euros)*";
- $\tilde{Y}_4$ ="*ville*".

	$\tilde{Y}_1$	$\tilde{Y}_2$	$\tilde{Y}_3$	$\tilde{Y}_4$
1	maison	65	75.000	Bruxelles
2	maison	115	115.000	Bruxelles
3	studio	40	42.000	Bruxelles
4	appartement	86	74.400	Bruxelles
5	maison	478	545.000	Bruxelles
6	appartement	140	147.500	Namur
7	studio	50	35.900	Namur
8	maison	230	123.000	Namur
9	villa	250	300.000	Namur
10	maison	300	138.000	Namur
11	appartement	90	130.150	Arlon
12	studio	60	44.000	Arlon
13	maison	204	80.000	Arlon
14	villa	374	375.000	Arlon
15	maison	240	350.000	Arlon

Nous allons agréger les individus en trois classes :
- $C_1 = \{x_k \mid \tilde{Y}_3(x_k) < 100.000\}$ l'ensemble des biens immobiliers dont le prix est inférieur à 100.000 €;
- $C_2 = \{x_k \mid 100.000 \leq \tilde{Y}_3(x_k) \leq 250.000\}$ l'ensemble des biens immobiliers dont le prix est compris entre 100.000 et 250.000 €;
- $C_3 = \{x_k \mid \tilde{Y}_3(x_k) > 250.000\}$ l'ensemble des biens immobiliers dont le prix est supérieur à 250.000 €.

Nous obtenons alors le tableau de données symboliques suivant :

	Y_1	Y_2	Y_3	Y_4
C_1	(maison $\frac{1}{3}$,studio $\frac{1}{2}$,appart. $\frac{1}{6}$)	{40,50,60,65,86,204}	[35.900,80.000]	{A,B,N}
C_2	(maison $\frac{3}{5}$,appart. $\frac{2}{5}$)	{90,115,140,230,300}	[115.000,147.000]	{A,B,N}
C_3	(maison $\frac{1}{2}$,villa $\frac{1}{2}$)	{240,250,374,478}	[300.000,545.000]	{A,B,N}

où Y_1 est une variable modale, Y_2 est une variable multivaluée ordinale, Y_3 est une variable de type intervalle et Y_4 est une variable multivaluée catégorique et où nous avons noté A pour Arlon, B pour Bruxelles et N pour Namur.

2.7 Dissimilarités entre objets symboliques

Dans cette section, nous nous intéresserons uniquement aux dissimilarités entre deux objets décrits par des variables multivaluées et modales.

2.7.1 Le cas des variables multivaluées

Intéressons-nous d'abord aux variables multivaluées. Considérons $E = \{x_1, \ldots, x_n\}$ un ensemble de n objets décrits par p variables multivaluées catégoriques $Y_1, \ldots, Y_p$ d'espaces d'observation $\mathcal{Y}_1, \ldots, \mathcal{Y}_p$ respectivement. $Y_j(x_k)$ est donc un ensemble de catégories de $\mathcal{Y}_j$.

Nous pouvons transformer la matrice originale $\underline{X} = (Y_j(x_k))_{n \times p}$ en une matrice des fréquences $\tilde{X}$.

Par définition, $Y_j(x_k)$ est l'ensemble des catégories que prend la variable multivaluée catégorique Y_j pour décrire l'objet $x_k \in E$. Notons m_j le nombre de modalités que peut prendre la variable Y_j, c'est-à-dire $m_j = |\mathcal{Y}_j|$. La valeur de distribution $q_{j,x_k}(c_s)$ associée à la catégorie c_s $(s = 1, \ldots, m_j)$ de Y_j est donnée par

$$q_{j,x_k}(c_s) = \begin{cases} \frac{1}{|Y_j(x_k)|} & \text{si } \{c_s\} \in Y_j(x_k) \\ 0 & \text{sinon.} \end{cases}$$

Ainsi, chaque objet symbolique $x_k \in E$ peut être représenté par un vecteur de dimension $m_1 + \ldots + m_p$, c'est-à-dire

$$x_k = ((q_{1,x_k}(c_1), \ldots, q_{1,x_k}(c_{m_1})), \ldots, (q_{p,x_k}(c_1), \ldots, q_{p,x_k}(c_{m_p}))).$$

De cette manière, nous obtenons la matrice des fréquences $\tilde{X}$ de dimension $n \times (m_1 + \ldots + m_p)$ suivante :

	Y_1			$\cdots$	Y_p		
	1	$\cdots$	m_1	$\cdots$	1	$\cdots$	m_p
1	$q_{1,x_1}(c_1)$	$\cdots$	$q_{1,x_1}(c_{m_1})$	$\cdots$	$q_{p,x_1}(c_1)$	$\cdots$	$q_{p,x_1}(c_{m_p})$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
k	$q_{1,x_k}(c_1)$	$\cdots$	$q_{1,x_k}(c_{m_1})$	$\cdots$	$q_{p,x_k}(c_1)$	$\cdots$	$q_{p,x_k}(c_{m_p})$
$\vdots$	$\vdots$		$\vdots$		$\vdots$		$\vdots$
n	$q_{1,x_n}(c_1)$	$\cdots$	$q_{1,x_n}(c_{m_1})$	$\cdots$	$q_{p,x_n}(c_1)$	$\cdots$	$q_{p,x_n}(c_{m_p})$

où les $q_{j,x_k}(c_i)$ sont tels que $\sum_{j=1}^{p} \sum_{i=1}^{m_j} q_{j,x_k}(c_i) = p$, $\forall k = 1, \ldots, n$.

Exemple 2.7.1
Considérons la matrice de données symboliques suivante :

x_k	*couleur*
x_1	*{rouge}*
x_2	*{bleu, vert, rouge}*
x_3	*{bleu, jaune}*

où $Y(x_k)$ ="la couleur de l'objet x_k" est une variable multivaluée catégorique d'espace d'observation $\mathcal{Y}$ ={bleu, jaune, rouge, vert}.

Nous codons alors cette matrice de la façon suivante :

x_k	*couleur*			
	bleu	*jaune*	*rouge*	*vert*
x_1	*0*	*0*	*1*	*0*
x_2	*1/3*	*0*	*1/3*	*1/3*
x_3	*1/2*	*1/2*	*0*	*0*

Grâce à cette matrice des fréquences, nous pouvons définir une mesure de dissimilarité sur E à partir de p indices de dissimilarité sur les $\mathcal{B}_j$. Considérons

$$\begin{array}{rccl} \delta_j : & \mathcal{B}_j \times \mathcal{B}_j & \to & \mathbb{R}^+ \\ & (x_{kj}, x_{lj}) & \rightsquigarrow & \delta_j(x_{kj}, x_{lj}). \end{array}$$

Notons $x_{kj}^{(i)}$ la fréquence prise par la variable j pour l'individu x_k concernant la modalité i. Nous pouvons alors définir trois distances :

1. La distance L_1 :
$$\delta_j(x_{kj}, x_{lj}) = \sum_{i=1}^{|\mathcal{Y}_j|} |x_{kj}^{(i)} - x_{lj}^{(i)}|.$$

2. La distance L_2 :
$$\delta_j(x_{kj}, x_{lj}) = \sum_{i=1}^{|\mathcal{Y}_j|} (x_{kj}^{(i)} - x_{lj}^{(i)})^2.$$

3. La distance de De Carvalho :

$$\delta_j(x_{kj}, x_{lj}) = \sum_{i=1}^{|\mathcal{Y}_j|} (\gamma x_{kj}^{(i)} + \gamma' x_{lj}^{(i)})$$

où

- $\gamma = \begin{cases} 1 & \text{si } x_{kj} \text{ prend la modalité } i \text{ et pas } x_{lj} \\ 0 & \text{sinon} \end{cases}$
- $\gamma' = \begin{cases} 1 & \text{si } x_{lj} \text{ prend la modalité } i \text{ et pas } x_{kj} \\ 0 & \text{sinon.} \end{cases}$

Nous pouvons, grâce à ces trois distances, nous ramener à une dissimilarité sur l'ensemble E des individus en définissant :

$$\begin{aligned} d: \quad E \times E &\to \mathbb{R}^+ \\ (x_k, x_l) &\rightsquigarrow d(x_k, x_l) = \left(\sum_{j=1}^{p} \delta_j^2(x_{kj}, x_{lj}) \right)^{1/2} \end{aligned}$$

où δ_j est une des dissimilarités définies précédemment.

2.7.2 Le cas des variables modales

Considérons $E = \{x_1, \ldots, x_n\}$ un ensemble de n objets décrits par p variables modales $Y_1, \ldots, Y_p$ d'espaces d'observation $\mathcal{Y}_1, \ldots, \mathcal{Y}_p$ respectivement.

Nous avons vu que, dans le cas de variables modales, la matrice de données symboliques $\underline{X} = (Y_j(x_k))_{n \times p} = (x_{kj})_{n \times p}$ était telle que chaque cellule x_{kj} contenait un histogramme.

Notons m_j le nombre de modalités que peut prendre la variable Y_j. Chaque objet symbolique $x_k \in E$ peut dès lors être représenté par un vecteur de dimension $m_1 + \ldots + m_p$, c'est-à-dire

$$x_k = ((q_{1,x_k}(c_1), \ldots, q_{1,x_k}(c_{m_1})), \ldots, (q_{p,x_k}(c_1), \ldots, q_{p,x_k}(c_{m_p})))$$

où $q_{j,x_k}(c_s)$ représente la valeur de la distribution associée à la catégorie c_s $(s = 1, \ldots, m_j)$ de Y_j.

De cette manière, nous pouvons obtenir la même matrice $\tilde{X}$ que celle définie pour les variables multivaluées catégoriques et nous pouvons définir les trois mêmes distances.

Chapitre 3

La classification automatique

3.1 Introduction

Le problème qui nous intéresse est celui de décomposer un ensemble d'individus décrits par des variables en un certain nombre de classes.

Une classe est un groupe homogène d'individus ayant des caractéristiques communes. Les objets dans une même classe doivent donc être similaires entre eux et différents des objets des autres classes.

Supposons un ensemble composé de n individus $\Omega = \{x_1, \ldots, x_n\}$ décrits par p variables $Y_1, \ldots, Y_p$.

Nous recherchons une partition de Ω en k classes (k fixé a priori).
Notons $P = \{C_1, \ldots, C_k\}$ et $\mathcal{P}_k$ l'ensemble de toutes les partitions de Ω en k classes.

Pour chaque partition $P \in \mathcal{P}_k$, nous associons le critère de classification suivant :

$$W : \mathcal{P}_k \to \mathbb{R} : P \rightsquigarrow W(P, k).$$

La partition optimale $P^* = \{C_1^*, \ldots, C_k^*\}$ sera telle que

$$W(P^*, k) = \min_{P \in \mathcal{P}_k} W(P, k) \text{ ou } W(P^*, k) = \max_{P \in \mathcal{P}_k} W(P, k).$$

Les différentes méthodes de classification se différencient par le critère utilisé qui se base le plus souvent sur une distance (ou dissimilarité) entre les objets.

3.2 Les structures classificatoires

Les classes $C_1, \ldots, C_k$ d'une classification $C = (C_1, \ldots, C_k)$ de Ω peuvent avoir ou non une certaine structure. Nous allons exposer les plus courantes.

3.2.1 Les recouvrements

Un recouvrement de Ω est un ensemble $R = \{C_1, \ldots, C_k\}$ de parties non vides de Ω dont la réunion forme Ω.

Définition : $R = \{C_1, \ldots, C_k\}$ est un recouvrement si et seulement si

1. $\forall l \in \{1, \ldots, k\}$, $C_l \neq \emptyset$;
2. $\cup_{l=1}^{k} C_l = \Omega$.

Exemple 3.2.1
Considérons un ensemble de neuf points décrits par deux variables. Un des recouvrements possibles de ces points par deux classes est donné à la figure 3.1.

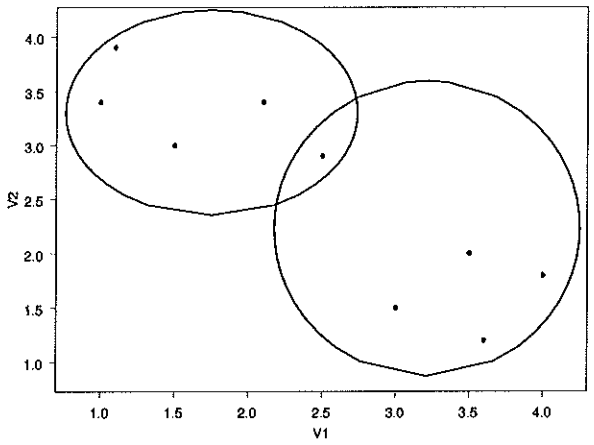


figure 3.1: Exemple de recouvrement

3.2.2 Les partitions

Une partition d'un ensemble Ω est un ensemble $P = \{C_1, \ldots, C_k\}$ de parties non vides de Ω d'intersections vides deux à deux et dont la réunion forme Ω.

Une partition est donc un recouvrement ayant la propriété suivante :

$$\forall l, m \in \{1, \ldots, k\}, l \neq m, C_l \cap C_m = \emptyset.$$

Exemple 3.2.2
Reprenons l'exemple précédent. Une des partitions possibles de ces points en deux classes est donnée à la figure 3.2.

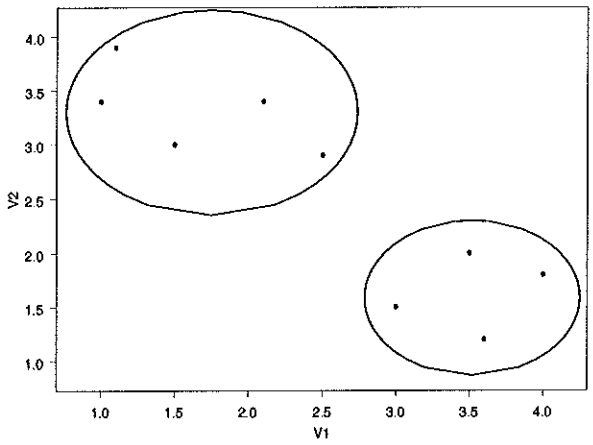


figure 3.2: Exemple de partition

3.2.3 Les hiérarchies

Une hiérarchie permet de représenter l'ensemble Ω des individus par un ensemble de partitions emboîtées.

Soient Ω un ensemble fini et H un ensemble de parties (appelées paliers) non vides de Ω.

Une hiérarchie H sur Ω est définie par :

1. $\Omega \in H$, c'est-à-dire que le palier le plus haut contient tous les individus ;
2. $\forall x_i \in \Omega$, $\{x_i\} \in H$, c'est-à-dire que chaque singleton appartient à H ;
3. $\forall h, h' \in H$, nous avons $h \cap h' \neq \emptyset \Rightarrow h \subset h'$ ou $h' \subset h$.

Une hiérarchie est visualisée par un arbre hiérarchique, appelé également dendogramme.

Exemple 3.2.3
Le dendogramme associé à l'exemple précédent est donné à la figure 3.3.

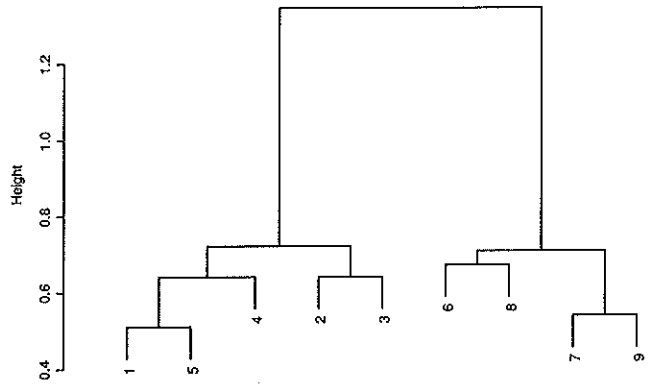


figure 3.3: Exemple de hiérarchie

3.3 Les méthodes de classification

Les méthodes de classification sont nombreuses et ont pour but de construire une partition ou une suite de partitions emboîtées. Les classes ainsi formées doivent être les plus homogènes possibles. Il existe principalement deux grands types de méthodes, les méthodes de partitionnement et les méthodes hiérarchiques.

3.3.1 Les méthodes hiérarchiques

Celles-ci peuvent être soit ascendantes (ou agglomératives) ou soit descendantes.

Les classifications hiérarchiques ascendantes partent de l'ensemble des classes à un seul élément et à chaque pas réunissent les deux classes les plus ressemblantes au sens du critère choisi jusqu'à obtenir la classe contenant tous les éléments. Elles construisent alors une suite de partitions emboîtées appelée hiérarchie.
Nous présenterons quatre méthodes hiérarchiques agglomératives par la suite.

Les méthodes hiérarchiques descendantes considèrent qu'au départ tous les individus appartiennent à une même classe. A chaque étape de l'algorithme, une nouvelle classe est créée par un processus de division, jusqu'à ce que tous les éléments se situent dans des classes différentes.

3.3.2 Les méthodes non-hiérarchiques

Ces méthodes fonctionnent de la manière suivante. Elles considèrent une partition initiale des individus et les déplacent d'un groupe à l'autre de manière à obtenir une partition qui, au fil des itérations, améliore un critère mathématique.

La méthode des hypervolumes et la méthode des nuées dynamiques sont des méhodes non-hiérarchiques. La première sera détaillée à la fin de ce chapitre tandis que la seconde sera étudiée dans le chapitre suivant.

3.4 Les méthodes hiérarchiques agglomératives

3.4.1 Méthode du lien simple [9]

A chaque pas, la distance entre deux classes est définie par la plus petite distance entre deux points de chacune des deux classes.

La distance entre deux classes C_i et C_j d'une partition est donc définie par :

$$d(C_i, C_j) = \min_{\substack{x \in C_i \\ y \in C_j}} d(x, y).$$

Cette méthode a l'avantage d'être simple, rapide et invariante par rapport aux transformations monotones des dissimilarités.
Cependant, elle présente les défauts d'être peu robuste et d'engendrer un effet de chaînage. Mais cet effet peut être un avantage lorsqu'il s'agit de retrouver des classes allongées.

3.4.2 Méthode du lien complet [9]

A chaque pas, la distance entre deux classes est définie par la distance maximum entre deux points de chacune des deux classes.

La distance entre deux classes C_i et C_j d'une partition est donc définie par :

$$d(C_i, C_j) = \max_{\substack{x \in C_i \\ y \in C_j}} d(x, y).$$

Les principaux avantages de cette méthode sont sa simplicité et son invariance par rapport aux transformations monotones des dissimilarités. Cependant, elle est peu robuste et biaisée par rapport aux classes hypersphériques, c'est-à-dire qu'elle a tendance à retrouver des classes hypersphériques quelle que soit la structure présente dans les données.

3.4.3 Méthode du centroïde [9]

Cette méthode est basée sur la notion de centre de gravité. Notons n_l le nombre d'individus présents dans la classe C_l. Le centroïde de la classe C_l est défini par

$$g^{(l)} = (g_1^{(l)}, \ldots, g_p^{(l)})$$

où $g_j^{(l)} = \frac{1}{n_l} \sum_{i=1}^{n_l} x_{ij}^{(l)}$.

La distance entre deux groupes sera la distance entre leurs centroïdes respectifs, c'est-à-dire :

$$d(C_i, C_j) = d(g^{(i)}, g^{(j)}).$$

Le principe de la méthode est de regrouper à chaque étape les deux groupes dont les centroïdes sont les plus proches.

Les deux principaux inconvénients de cette méthode sont qu'elle est biaisée par rapport aux classes hypersphériques et qu'elle perd les caractéristiques des petits groupes.

3.4.4 Méthode de Ward [6]

La méthode de Ward est une méthode hiérarchique basée sur le critère des moindres carrés.

Nous commençons par définir le centroïde de la classe C_l.

$$g^{(l)} = (g_1^{(l)}, \ldots, g_p^{(l)})$$

où $g_j^{(l)} = \frac{1}{n_l} \sum_{i=1}^{n_l} x_{ij}^{(l)}$ et n_l est le nombre d'individus dans la classe C_l.

Nous définissons ensuite pour chaque classe une erreur qui est équivalente à un indice de dispersion.

$$e_l^2 = \sum_{i=1}^{n_l} \sum_{j=1}^{p} (x_{ij}^{(l)} - g_j^{(l)})^2 = \sum_{i=1}^{n_l} \|x_i^{(l)} - g^{(l)}\|^2$$

où $(x_{ij}^{(l)} - g_j^{(l)})$ est la déviation de chaque point de la classe C_l par rapport au centroïde de la classe.

L'erreur associée à une partition sera alors

$$E_k^2 = \sum_{l=1}^{k} e_l^2,$$

ce qui n'est rien d'autre que le critère des moindres carrés.

Règle de regroupement :
Nous fusionnons les classes pour lesquelles ΔE_{rs}^2 est minimum où ΔE_{rs}^2 est l'accroissement de l'erreur E^2 lorsque les classes r et s sont regroupées.

Si les classes r et s sont fusionnées en la classe t, nous avons que

$$\Delta E_{rs}^2 = e_t^2 - e_r^2 - e_s^2.$$

Nous pouvons aussi montrer que

$$\Delta E_{rs}^2 = \frac{n_r n_s}{n_r + n_s} \sum_{j=1}^{p} (g_j^{(r)} - g_j^{(s)})^2.$$

Remarquons que le critère des moindres carrés E_k^2 augmente lorsque le nombre de classes diminue.

A chaque étape, le critère de la variance est optimisé mais cela ne garantit pas que la partition finale en k classes sera celle pour laquelle le critère de la variance est minimum.
Cette méthode est également biaisée par rapport aux classes hypersphériques.

3.5 La méthode des hypervolumes [12]

Contrairement aux quatre méthodes hiérarchiques que nous venons de voir, la méthode des hypervolumes est non-hiérarchique et est basée sur un modèle statistique.

3.5.1 Processus de Poisson

Rappelons tout d'abord ce qu'est un processus de Poisson.

Définitions :
N est un **processus de Poisson** sur $E \subset \mathbb{R}^p$ si et seulement si

1. $\forall A_1, \ldots, A_k \subset E$, disjoints 2 à 2, les variables aléatoires $N(A_1), \ldots, N(A_k)$ sont indépendantes (où $N(A_i)$ représente le nombre de points dans A_i) ;
2. $\forall A \subset E$ et $k \geq 0$, $P(N(A) = k) = e^{-\mu(A)} \frac{(\mu(A))^k}{k!}$ où μ est une mesure qui donne une masse finie à tout ensemble borné.

N est un **processus de Poisson homogène (ou stationnaire)** sur $E \subset \mathbb{R}^p$ si et seulement si

1. N est un processus de Poisson sur E ;
2. $\mu(A) = \rho m(A)$ où
   - $m(A)$ est la mesure de Lebesgue de A ;
   - $\rho \in \mathbb{R}_0^+$ est l'intensité du processus.

3.5.2 Le modèle statistique

Le modèle statistique sur lequel repose la méthode des hypervolumes contient les hypothèses suivantes :

- les variables aléatoires qui comptent les nombres de points dans les régions disjointes sont indépendantes ;
- le nombre moyen de points dans chaque région est proportionnel à la mesure de Lebesgue de cette région.

Nous supposons donc que les points sont générés par un processus de Poisson homogène dans $D \subseteq \mathbb{R}^p$ où $D = \bigcup_{i=1}^{k} D_i$ et les domaines D_i sont disjoints avec k fixé a priori.

3.5.3 Solution statistique

Notre but est d'estimer les frontières des k domaines $D_1, \ldots, D_k$. Pour cela, nous utiliserons la fonction de vraisemblance.

Soit $x = (x_1, \ldots, x_n)$ le vecteur des réalisations du processus dans $D = \bigcup_{i=1}^{k} D_i$. La fonction de vraisemblance est

$$L(D, x) = f_D(x) = \left(\frac{1}{m(D)}\right)^n \prod_{i=1}^{n} I_D(x_i)$$

où

- $m(D)$ est la mesure de Lebesgue du domaine D ;
- $I_D(x_i)$ est la fonction indicatrice de D.

Le domaine D pour lequel la vraisemblance est maximale est, parmi ceux qui contiennent tous les points, celui dont la mesure de Lebesgue est minimale.

Pour éviter d'obtenir des solutions triviales pour ce problème, nous devons imposer une contrainte supplémentaire sur la structure de D. Cette contrainte impose que les domaines $D_1, \ldots, D_k$ doivent être convexes.

Le maximum de la fonction de vraisemblance sera alors atteint par la partition pour laquelle la somme des mesures de Lebesgue des enveloppes convexes des k domaines est minimale.

3.5.4 Le critère des hypervolumes

Considérons un ensemble $\Omega = \{x_1, \ldots, x_n\}$ de points engendrés par un processus de Poisson homogène dans $D = \bigcup_{i=1}^{k} D_i$ où les D_i sont convexes et disjoints.

Nous recherchons une partition de Ω en k classes disjointes $C_1, \ldots, C_k$ où k est fixé a priori.

Le critère des hypervolumes est défini par

$$\begin{aligned} W : \mathcal{P}_k &\to \mathbb{R}^+ \\ P = \{C_1, \ldots, C_k\} &\rightsquigarrow W(P, k) = \sum_{i=1}^{k} m(H(C_i)) \end{aligned}$$

où
- $\mathcal{P}_k$ est l'ensemble des partitions possibles de l'ensemble Ω en k classes ;
- $H(C_i)$ est l'enveloppe convexe des points appartenant à la classe C_i ;
- m est la mesure de Lebesgue multidimensionnelle.

Le problème est donc de maximiser la fonction de vraisemblance $f_D(x)$ pour déterminer la partition optimale $P^* \in \mathcal{P}_k$. Ce qui revient à chercher $P^* \in \mathcal{P}_k$ tel que

$$W(P^*, k) = \min_{P \in \mathcal{P}_k} W(P, k) = \min \sum_{i=1}^{k} m(H(C_i)).$$

Pratiquement, dans $\mathbb{R}^2$, nous recherchons la partition en k classes telles que la somme des aires de leurs enveloppes convexes soit minimale. De manière générale, dans $\mathbb{R}^p$, cela revient à rechercher la partition en k classes telles que la somme des hypervolumes de leurs enveloppes convexes soit minimale.

3.6 Classification des données symboliques

Les cinq méthodes de classifications automatique que nous venons de voir s'appliquent à des données classiques et peuvent être adaptées pour les données symboliques. Cependant, il existe aussi des méthodes de classification spécifiques aux données symboliques. Nous pouvons citer par exemple Sclust, la méthode divisive de Chavent (DIV), les arbres de clustering,....

En quelques mots, la méthode divisive de Chavent est une méthode hiérarchique qui commence avec une classe comprenant tous les objets et qui procède par divisions successives de chaque classe. A chaque étape, une classe est divisée en deux groupes selon une question binaire. Cette question binaire induit la meilleure partition en deux classes selon une extension du critère de l'inertie. L'algorithme s'arrête après $k - 1$ divisions, où k est le nombre de classes fixées par l'utilisateur.

Sclust est une adaptation de la méthode des nuées dynamiques classique aux données symboliques. Nous présenterons cette méthode dans le chapitre suivant.

Chapitre 4

La méthode des nuées dynamiques

4.1 Introduction

La méthode des nuées dynamiques a été introduite par E. Diday en 1972. Son but est de trouver, parmi l'ensemble de toutes les partitions possibles en k classes, une partition qui optimise un critère défini a priori.

Comme l'ensemble E des individus est fini, l'ensemble des partitions possibles est aussi fini et nous pourrions résoudre ce problème en énumérant toutes les partitions et en calculant le critère pour chacune d'entre elles. Mais cette recherche exhaustive est infaisable en pratique car si nous recherchons la meilleure partition en k classes d'un ensemble E contenant n individus, il faudrait analyser $\frac{k^n}{k!}$ partitions. Une façon de résoudre ce problème est de partir d'une solution acceptable et réalisable et de l'améliorer par itérations successives.

Avant de décrire la méthode des nuées dynamiques dans le cas classique et dans le cas de données symboliques, nous avons besoin d'introduire la notion d'inertie.

Il s'agit de la somme des distances au carré de chaque point au centre de gravité global du nuage de points.

- l'inertie intra-classes :

$$W = \sum_{l=1}^{k} \sum_{i=1}^{n_l} d^2(x_i, g^{(l)}) = \sum_{l=1}^{k} I_{g^{(l)}}(C_l).$$

Il s'agit de la somme des inerties de chaque classe à son centre de gravité.

- l'inertie inter-classes :

$$B = \sum_{l=1}^{k} n_l d^2(g^{(l)}, g).$$

Il s'agit de l'inertie du nuage formé par les centres de gravité $g^{(l)}$ de chaque classe pondérés par le nombre d'individus de la classe C_l par rapport au centre de gravité global g.

Ces trois inerties sont reliées par la relation suivante :

$$T = B + W.$$

Or, l'inertie totale T est indépendante de la partition. Cette relation nous permet donc d'affirmer que plus l'inertie inter-classes B est grande, plus l'inertie intra-classes W est petite.

4.2.3 Décomposition des inerties sur les classes et les variables

Nous allons maintenant décomposer ces trois inerties en fonction des classes et des variables.

L'inertie totale T peut aussi être définie par :

$$T = \sum_{l=1}^{k} \sum_{j=1}^{p} T_j^{(l)} \text{ où } T_j^{(l)} = \sum_{x_i \in C_l} (x_{ij} - g_j)^2.$$

$T_j^{(l)}$ représente l'écart pour la variable j des points de la classe C_l au centre de gravité global g_j.

L'inertie intra-classes W peut s'écrire :

$$W = \sum_{l=1}^{k} \sum_{j=1}^{p} W_j^{(l)} \text{ où } W_j^{(l)} = \sum_{x_i \in C_l} (x_{ij} - g_j^{(l)})^2.$$

$W_j^{(l)}$ est l'inertie de la classe C_l par rapport à son centre de gravité $g_j^{(l)}$ pour la variable j.

L'inertie inter-classes B peut se décomposer de la manière suivante :

$$B = \sum_{l=1}^{k} \sum_{j=1}^{p} B_j^{(l)} \text{ où } B_j^{(l)} = n_l (g_j^{(l)} - g_j)^2.$$

$B_j^{(l)}$ est l'écart, pour la variable j, du centre de gravité local $g_j^{(l)}$ de la classe C_l au centre de gravité global g_j.

Les inerties peuvent aussi se décomposer de manière additive sur les variables et sur les classes.

Notons $\forall i = 1, \ldots, p$

$$T_j = \sum_{l=1}^{k} T_j^{(l)} \qquad W_j = \sum_{l=1}^{k} W_j^{(l)} \qquad B_j = \sum_{l=1}^{k} B_j^{(l)}.$$

Nous obtenons alors

$$T = \sum_{j=1}^{p} T_j \qquad W = \sum_{j=1}^{p} W_j \qquad B = \sum_{j=1}^{p} B_j$$

où T_j, W_j, et B_j représentent la contribution de la variable j pour les inerties concernées.

Notons maintenant $\forall l = 1, \ldots, k$

$$T^{(l)} = \sum_{j=1}^{p} T_j^{(l)} \qquad W^{(l)} = \sum_{j=1}^{p} W_j^{(l)} \qquad B^{(l)} = \sum_{j=1}^{p} B_j^{(l)}.$$

Les différentes inerties s'écrivent alors

$$T = \sum_{l=1}^{k} T^{(l)} \qquad W = \sum_{l=1}^{k} W^{(l)} \qquad B = \sum_{l=1}^{k} B^{(l)}$$

où $T^{(l)}$, $W^{(l)}$ et $B^{(l)}$ expriment la contribution de la classe l pour les inerties concernées.

4.3 La méthode des nuées dynamiques dans le cas de données classiques [5]

4.3.1 Principe général

L'algorithme des nuées dynamiques se décompose en deux étapes. La première étape consiste à définir un représentant ou prototype pour chacune des classes et s'appelle l'étape de représentation. La seconde étape modifie la classe d'affectation de chacun des individus de E et s'appelle l'étape d'affectation.

Commençons par définir un mode de représentation des classes. Le représentant d'une classe peut être une droite, un ensemble de points ou encore un centre de gravité. Dans le cadre de la méthode des nuées dynamiques, les représentants des classes sont appelés noyaux.

L'algorithme se déroule de la manière suivante :
- L'algorithme commence par choisir k noyaux estimés ou tirés au hasard parmi une famille de noyaux admissibles, appelée espace de représentation et notée $\mathcal{L}$;
- Ensuite, les points sont affectés au noyau le plus proche.
  Une nouvelle partition en k classes est ainsi constituée et les nouveaux noyaux associés à cette partition sont calculés ;
- Le procédé est alors recommencé avec les nouveaux noyaux.

Sous certaines conditions portant sur les fonctions qui permettent d'affecter les points aux classes et de calculer les noyaux, l'algorithme fait décroître un critère W qui mesure l'adéquation entre les classes et leur noyau respectif, c'est-à-dire la "ressemblance" des noyaux à leur classe.

Considérons D une mesure d'adéquation du noyau L_l à la classe C_l. Une petite valeur de D va exprimer une bonne adéquation entre le noyau L_l et la classe C_l.

Le critère s'exprime de la manière suivante :

$$\begin{aligned} W : \ \mathcal{P}_k \times \mathcal{L}_k &\to \mathbb{R}^+ \\ (P, L) &\rightsquigarrow W(P, L) = \sum_{l=1}^{k} D(L_l, C_l) \end{aligned}$$

où
- $\mathcal{L}_k$ est l'ensemble des k-uples de noyaux $L = (L_1, \ldots, L_k)$;
  $\mathcal{L}_k = \Omega^k$ si $L_l \in \Omega$, c'est-à-dire si chaque noyau est un point de Ω ;
  $\mathcal{L}_k = \mathbb{R}^k$ si L_l est le centre de gravité de la classe C_l ;
- $\mathcal{P}_k$ est l'ensemble des partitions $P = (C_1, \ldots, C_k)$ en k classes de Ω.

La diminution de la valeur du critère à chaque itération de l'algorithme exprime ainsi l'augmentation de l'adéquation entre les classes et leurs noyaux.

4.3.2 Aspect formel

Le but de la méthode des nuées dynamiques est de trouver un couple $(P^*, L^*) \in \mathcal{P}_k \times \mathcal{L}_k$ tel que

$$W(P^*, L^*) = \min_{\substack{P \in \mathcal{P}_k \\ L \in \mathcal{L}_k}} W(P, L)$$

où $\mathcal{P}_k$ est l'ensemble des partitions possibles en k classes et $\mathcal{L}_k$ est un espace de représentation.

La minimisation de ce critère est réalisée en deux étapes successives. La première est l'étape de représentation et la seconde est l'étape d'affectation. Ces deux étapes sont exécutées itérativement jusqu'à obtenir la convergence du critère.

La méthode des nuées dynamiques consiste à :

1. Choisir un espace de représentation $\mathcal{L}_k$;
2. Définir un critère $W : \mathcal{P}_k \times \mathcal{L}_k \to \mathbb{R}^+ : (P, L) \rightsquigarrow W(P, L)$ qui mesure l'adéquation entre une partition $P \in \mathcal{P}_k$ et sa représentation $L \in \mathcal{L}_k$;
3. Chercher simultanément une partition $P \in \mathcal{P}_k$ et sa représentation $L \in \mathcal{L}_k$ de sorte que P et L aient la meilleure adéquation au sens du critère W.

Ce problème peut être résolu grâce à l'algorithme des nuées dynamiques. Celui-ci utilise itérativement
- une fonction de représentation $g : \mathcal{P}_k \to \mathcal{L}_k$;
- une fonction d'affectation $f : \mathcal{L}_k \to \mathcal{P}_k$

jusqu'à obtenir la convergence du critère.

La fonction de représentation g permet de calculer les k noyaux à partir de la partition P en k classes, tandis que la fonction d'affectation f construit la partition P en k classes en affectant chaque individu au noyau dont il est le plus proche.

Initialisation
L'initialisation de l'algorithme se fait soit à partir d'une partition initiale $P^0 \in \mathcal{P}_k$ ou soit à partir d'une représentation $L^0 \in \mathcal{L}_k$, estimées ou tirées au hasard.

Nous pouvons alors définir une suite $u_n = W(v_n)$ avec $v_n = (P^n, L^n)$ où :
- $P^n \in \mathcal{P}_k$ est la partition en k classes obtenue à l'itération n ;
- $L^n = g(P^n) \in \mathcal{L}_k$ est la représentation de la partition P^n obtenue à l'aide de la fonction g.

La suite u_n est donc la valeur du critère associée à v_n.

Convergence
Nous avons que $f(L^n) = P^{n+1}$ et $g(P^{n+1}) = L^{n+1}$. Si les fonctions d'affectation f et de représentation g sont bien choisies, alors
- les suites v_n et u_n sont convergentes ;
- la suite u_n est décroissante.

Ceci nous permet de conclure que l'algorithme fait décroître la valeur du critère à chaque itération jusqu'à stabilisation.

4.3.3 Le cas du centre de gravité

Nous allons décrire dans cette section les différentes parties de l'algorithme des nuées dynamiques dans le cas où les noyaux ou représentants de chaque classe sont définis par les centres de gravité.

Espace de représentation
L'espace des individus est l'espace $\mathbb{R}^p$ ainsi que l'espace de représentation $\mathcal{L}$ d'une classe.

La mesure d'adéquation dans ce cas-ci est l'inertie de la partition considérée par rapport à un point de $\mathbb{R}^p$. C'est-à-dire,

$$\begin{aligned} D : \mathcal{P} \times \mathcal{L} &\to \mathbb{R}^+ \\ (A, x) &\rightsquigarrow D(A, x) = \sum_{a \in A} d^2(a, x) = I_x(A) \end{aligned}$$

où d est une distance euclidienne.

La fonction de représentation g
La fonction de représentation g qui à toute partition $P = (C_1, \ldots, C_k)$ associe sa représentation $L = (L_1, \ldots, L_k)$ est définie par :

$$\begin{array}{rccc} g : & \mathcal{P}_k & \to & \mathcal{L}_k \\ & (C_1, \ldots, C_k) & \rightsquigarrow & (g^{(1)}, \ldots, g^{(k)}) \end{array}$$

où $g^{(l)}$ est le centre de gravité de la classe l.

Rappelons que, par le théorème de Huygens, le centre de gravité est le point par rapport auquel l'inertie du nuage de points est minimale.

La fonction d'affectation
La fonction d'affectation f est définie par

$$\begin{array}{rccc} f : & \mathcal{L}_k & \to & \mathcal{P}_k \\ & (g^{(1)}, \ldots, g^{(k)}) & \rightsquigarrow & (C_1, \ldots, C_k) \end{array}$$

où $C_l = \{x \in \mathbb{R}^p : d(x, g^{(l)}) \leq d(x, g^{(m)}), \forall m \in \{1, \ldots, k\}$ et $l < m$ en cas d'égalité$\}$.

La classe C_l est donc composée des individus les plus proches, au sens de la métrique choisie, de son centre de gravité.

Le problème d'optimisation
L'objectif est de trouver le couple $(P^*, L^*) \in \mathcal{P}_k \times \mathcal{L}_k$ qui minimise le critère d'adéquation W entre une partition $P = (C_1, \ldots, C_k)$ et sa représentation $L = (g^{(1)}, \ldots, g^{(k)})$. Ce critère est défini par

$$W(P, L) = \sum_{l=1}^{k} D(C_l, g^{(l)}) = \sum_{l=1}^{k} \sum_{x_i \in C_l} d^2(x_i, g^{(l)})$$

où $g^{(l)}$ est le centre de gravité de la classe l.

Notons W_l l'inertie de la classe C_l par rapport à son centre de gravité $g^{(l)}$. Le critère peut alors s'écrire sous la forme

$$W(P, L) = \sum_{l=1}^{k} W_l = W.$$

C'est-à-dire que le critère que la méthode des nuées dynamiques cherche à minimiser, dans le cas où les noyaux sont les k centres de gravité, est l'inertie intra-classes W.

Par la relation $T = W + B$, ceci revient également à maximiser l'inertie inter-classes B.

Convergence
Définissons les deux suites suivantes :

$$v_n = (P^n, L^n) \text{ et } u_n = W(v_n).$$

Ces suites vérifient les propriétés suivantes :
- La suite v_n est stationnaire ;
- La suite u_n converge en décroissant.

4.4 La méthode des nuées dynamiques dans le cas de données symboliques

Nous allons uniquement décrire la méthode des nuées dynamiques dans le cas de données symobliques décrites par des variables multivaluées catégoriques et par des variables modales.

Ces deux cas sont fort semblables, c'est pourquoi nous les décrirons simultanément.

Considérons un ensemble $E = \{x_1, \ldots, x_n\}$ contenant n objets symboliques décrits par p variables multivaluées ou modales $Y_1, \ldots, Y_p$ où Y_j a pour espace d'observation $\mathcal{Y}_j$.

La première phase de la méthode des nuées dynamiques consiste à définir une représentation des classes d'objets. Ici, nous proposons de construire les prototypes en résumant toute l'information contenue dans les différentes classes sur les objets symboliques puisque c'est de cette manière que le programme Sclust fonctionnne.

Dans les deux cas qui nous préoccupent, chaque prototype associé à une classe sera représenté comme un objet symbolique de type modal. La description du prototype est donnée par les distributions de fréquence (ou de probabilité) associées aux p variables.

Rappelons qu'un objet symbolique x_k décrit par p variables multivaluées catégoriques ou modales était représenté dans le chapitre 2 par un vecteur de dimension $m_1 + \ldots + m_p$ où m_j représente le nombre de modalités que peut prendre la variable Y_j, c'est-à-dire

$$x_k = ((q_{1,x_k}(c_1), \ldots, q_{1,x_k}(c_{m_1})), \ldots, (q_{p,x_k}(c_1), \ldots, q_{p,x_k}(c_{m_p}))).$$

Le prototype de la classe C_l est alors construit de la manière suivante :

$$\left(\left(\frac{1}{n_l}\sum_{x_i\in C_l} q_{1,x_i}(c_1),\ldots,\frac{1}{n_l}\sum_{x_i\in C_l} q_{1,x_i}(c_{m_1})\right),\ldots,\left(\frac{1}{n_l}\sum_{x_i\in C_l} q_{p,x_i}(c_1),\ldots,\frac{1}{n_l}\sum_{x_i\in C_l} q_{p,x_i}(c_{m_p})\right)\right)$$

où n_l est le nombre d'individus dans la classe C_l.

La mesure d'adéquation entre les classes et leurs noyaux est définie par

$$D : \mathcal{P} \times \mathcal{L} \to \mathbb{R}^+$$

telle que

$$\forall A \in \mathcal{P} \text{ et } \forall x_i \in \mathcal{L}, \quad D(A, x_i) = \sum_{a\in A} d^2(a, x_i)$$

où d est une mesure de dissimilarité définie pour des objets symboliques de type modal.

La fonction de représentation g, qui à toute partition $P = (C_1, \ldots, C_k)$ associe sa représentation $L = (L_1, \ldots, L_k)$, est définie par

$$\begin{array}{rccc} g : & \mathcal{P}_k & \to & \mathcal{L}_k \\ & (C_1, \ldots, C_k) & \rightsquigarrow & (g^{(1)}, \ldots, g^{(k)}) \end{array}$$

où $g^{(l)}$ est le prototype de la classe C_l défini comme ci-dessus.

La fonction d'affectation f est définie par

$$\begin{array}{rccc} f : & \mathcal{L}_k & \to & \mathcal{P}_k \\ & (g^{(1)}, \ldots, g^{(k)}) & \rightsquigarrow & (C_1, \ldots, C_k) \end{array}$$

où $C_l = \{x \in E \mid d(x, g^{(l)}) \leq d(x, g^{(m)}), \forall m \in \{1, \ldots, k\} \text{ et } l < m \text{ en cas d'égalité}\}$. C'est-à-dire que C_l est la classe composée des objets symboliques les plus proches, au sens de la métrique choisie, de son prototype $g^{(l)}$.

Le problème d'optimisation est celui de chercher le couple $(P^*, L^*) \in \mathcal{P}_k \times \mathcal{L}_k$ minimisant le critère d'adéquation W entre la partition $P = (C_1, \ldots, C_k)$ et sa représentation $L = (g^{(1)}, \ldots, g^{(k)})$. Ce critère est défini par :

$$W(P, L) = \sum_{l=1}^{k} D(C_l, g^{(l)}) = \sum_{l=1}^{k} \sum_{x_i\in C_l} d^2(x_i, g^{(l)})$$

où $g^{(l)}$ est le prototype de la classe C_l.

Nous pouvons aussi écrire ce critère sous la forme

$$W(P, L) = \sum_{l=1}^{k} W_l$$

Chapitre 5

Les méthodes de détermination du nombre de classes

5.1 Introduction

Nous venons de décrire plusieurs méthodes permettant de partitionner l'ensemble des données en un nombre k de classes (k fixé a priori). L'un des problèmes les plus difficiles en matière de classification est le choix du nombre de groupes qui doivent résulter de la segmentation des données. Pour cela, il nous faut évaluer la qualité des partitions ainsi produites. De nombreux critères ont été proposés dans la littérature ; Milligan et Cooper en ont testés une trentaine en 1988. Nous allons voir les cinq meilleurs méthodes issues de leur classement ainsi qu'une méthode basée sur le critère des hypervolumes.

5.2 Les méthodes de Milligan et Cooper

5.2.1 La méthode de Calinski et Harabasz (M1) [4]

Le critère le plus prometteur est celui introduit par Calinski et Harabasz en 1974, aussi appelé "le critère de rapport de variances".

Considérons un ensemble $\Omega = \{x_1, \ldots, x_n\}$ de n individus décrits par p variables $Y_1, \ldots, Y_p$.

Supposons la matrice de données suivante :

$$\tilde{X} = \begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix}$$

Définissons :
- La matrice de l'inertie totale :

$$M_T = \sum_{i=1}^{n} (x_i - g)(x_i - g)'$$

où
- g est le centre de gravité du nuage de points ;
- x_i est le $i^{\text{ème}}$ individu.

- La matrice de l'inertie intra-classes :

$$M_W = \sum_{l=1}^{k} M_W^{(l)} = \sum_{l=1}^{k} \sum_{i=1}^{n_l} (x_i^{(l)} - g^{(l)})(x_i^{(l)} - g^{(l)})'$$

où
- k est le nombre de classes ;
- $x_i^{(l)}$ est le $i^{\text{ème}}$ individu de la classe l ;
- n_l est le nombre d'individus dans la classe l ;
- $g^{(l)} = \frac{1}{n_l} \sum_{i=1}^{n_l} x_i^{(l)}$ est le centre de gravité de la classe l.

- La matrice d'inertie inter-classes :

$$M_B = \sum_{l=1}^{k} \sum_{i=1}^{n_l} (g^{(l)} - g)(g^{(l)} - g)'$$

où
- k est le nombre de classes ;
- n_l est le nombre d'individus dans la classe l ;
- g est le centre de gravité du nuage de points ;
- $g^{(l)} = \frac{1}{n_l} \sum_{i=1}^{n_l} x_i^{(l)}$ est le centre de gravité de la classe l.

Remarquons aussi que $M_T = M_B + M_W$.

L'indice de Calinski et Harabasz est défini par :

$$CH = \frac{tr(M_B)/(k-1)}{tr(M_W)/(n-k)} = \frac{B/(k-1)}{W/(n-k)}$$

où la dernière égalité résulte du fait que $tr(M_B) = B$ et que $tr(M_W) = W$.

Puisque nous voulons que l'inertie intra-classes soit la plus petite possible pour avoir des groupes homogènes et que $T = B + W$, l'inertie inter-classes B devra être la plus grande possible.

Le nombre k de classes présentes dans les données sera donc déterminé par un maximum relatif ou absolu de l'indice ou par un écart important entre deux de ses valeurs successives.

5.2.2 La méthode de Duda et Hart (M2) [8]

Considérons un ensemble $\Omega = \{x_1, \ldots, x_n\}$ de n individus décrits par p variables $Y_1, \ldots, Y_p$.

Définissons

$$\begin{aligned} J_e(k) &= \sum_{l=1}^{k} \sum_{i=1}^{n_l} \sum_{j=1}^{p} (x_{ij}^{(l)} - g_j^{(l)})^2 \\ &= tr(M_W) \\ &= W \end{aligned}$$

qui est la somme des carrés des erreurs qui surviennent si les n individus sont représentés par les k centres des classes.

La méthode de Duda et Hart se base sur le test d'hypothèse suivant :
H_0 : Les points sont issus d'une seule population Normale de moyenne μ et de matrice de variance-covariance $\sigma^2 I$.
H_1 : Les points sont issus de deux populations Normales.

Règle de décision :
Nous rejetons l'hypothèse H_0 au niveau de signification α si

$$\frac{J_e(2)}{Je(1)} < 1 - \frac{2}{\pi p} - z_{1-\alpha} \sqrt{\frac{2(1 - \frac{8}{\pi^2 p})}{np}}$$

où $z_{1-\alpha}$ est le quantile d'ordre $1 - \alpha$ de la loi Normale.

Cette méthode ne s'applique qu'à des méthodes de classification hiérarchiques. Ce test est effectué à chaque niveau de la hiérarchie entre les deux groupes candidats à la fusion. Si nous acceptons l'hypothèse H_0, cela signifie qu'il faut fusionner les deux classes considérées. Si nous rejetons la fusion pour $k = k_0$, alors le nombre optimal de classes sera $k_0 + 1$.

A chaque niveau de la hiérarchie, nous calculerons donc

$$DH = \frac{-\frac{J_e(2)}{Je(1)} + 1 - \frac{2}{\pi p}}{\sqrt{\frac{2(1-\frac{8}{\pi^2 p})}{np}}}$$

où

- $J_e(1) = \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - g_j)^2$ correspond à l'inertie d'une classe C ;
- $J_e(2) = \sum_{l=1}^{2} \sum_{i=1}^{n_l} \sum_{j=1}^{p} (x_{ij}^{(l)} - g_j^{(l)})^2$ correspond à la somme des inerties de deux classes C_1 et C_2 telles que $C_1 \cup C_2 = C$.

Remarquons que $J_e(1) > J_e(2)$. Nous rejetterons donc H_0 lorsque $DH > z_{1-\alpha}$.

Plusieurs valeurs pour $z_{1-\alpha}$ sont utilisées. D'après Milligan et Cooper, $z_{1-\alpha} = 3.20$ convient. Par contre, Gordon a proposé $z_{1-\alpha} = 4$ de façon à obtenir de meilleurs résultats.

5.2.3 La méthode du C-index (M3) [13]

Définissons :

$$C(x_i, x_j) = \begin{cases} 1 \text{ si } x_i \text{ et } x_j \text{ sont dans la même classe } (i \neq j) \\ 0 \text{ sinon} \end{cases}$$

et

$$\begin{aligned} \Gamma &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} C(x_i, x_j) \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} C(x_i, x_j) \\ &= \sum_{l=1}^{k} \sum_{\substack{i,j=1 \\ i<j}}^{n_l} d(x_i^{(l)}, x_j^{(l)}) \end{aligned}$$

où d est une distance euclidienne, k est le nombre de classes et n_l est le nombre d'individus dans la classe l.

Pour déterminer le nombre de classes, nous utilisons la normalisation de Γ proposée par Dalrymphe-Arford. Nous obtenons alors l'indice

$$\text{index } C = \frac{\Gamma - \min \Gamma}{\max \Gamma - \min \Gamma}.$$

Si la partition considérée contient r dissimilarités entre paires d'individus :
- $\min \Gamma$ est défini comme la somme des r plus petites dissimilarités entre paires d'individus ;
- $\max \Gamma$ est défini comme la somme des r plus grandes dissimilarités entre paires d'individus.

Le nombre de classes est indiqué par la valeur minimale de l'indice.

Propriétés :

1. $C \geq 0$;
2. Idéalement, il faut que $C = 0$. De plus, $C = 0 \Leftrightarrow \Gamma = \min \Gamma$.

Remarquons aussi que cet indice est biaisé par rapport aux classes hypersphériques puisque la somme des distances entre les éléments d'une classe doit être la plus petite possible.

5.2.4 La méthode Gamma (M4) [1]

Supposons un ensemble composé de n individus $\Omega = \{x_1, \ldots, x_n\}$. Soit d la distance euclidienne.
Nous définissons

$$T_l(x_i, x_j) = \begin{cases} 0 & \text{si } x_i \text{ et } x_j \text{ sont dans la même classe} \\ 1 & \text{sinon.} \end{cases}$$

Si $T_l(x_i, x_j) = 0$, posons

$$n_l(x_i, x_j) = \sharp\{\{x_r, x_s\} : T_l(x_r, x_s) = 1 \text{ et } d(x_r, x_s) < d(x_i, x_j)\}$$

qui est le nombre de couples d'objets qui n'appartiennent pas à la même classe mais qui sont plus proches que x_i de x_j.

Nous pouvons alors définir l'indice α_l par

$$\alpha_l = \frac{\sum_{i<j} n_l(x_i, x_j)}{\max \sum_{i<j} n_l(x_i, x_j)}$$

où le maximum est pris sur toutes les partitions possibles en k classes en gardant le même nombre d'objets par classe, et les sommes sont effectuées sur les paires d'objets $\{x_i, x_j\}$ qui appartiennent à la même classe.

Cet indice représente la proportion de paires d'objets qui sont dans de mauvais groupes, dans le sens où deux objets appartenant à une même classe doivent être plus proches entre eux que deux objets des classes différentes.

En effet, le numérateur représente la somme sur toutes les paires d'objets appartenant à la même classe du nombre d'objets appartenant à des classes différentes et qui sont plus proches l'un de l'autre que les paires d'objets considérés sur la somme.

L'indice γ peut ainsi être défini par

$$\gamma = 1 - 2\alpha_l.$$

Notons que $-1 \leq \gamma \leq 1$.
Si aucun objet n'est mal placé, alors $\alpha_l = 0$ et $\gamma = 1$. Si, par contre, un maximum d'objets est mal placé, alors $\alpha_l = 0$ et $\gamma = -1$.

La méthode Gamma cherche la valeur maximale de l'indice, celle-ci devant être la plus proche possible de 1 car pour cette valeur, la partition est dite "parfaite".

5.2.5 La méthode de Beale (M5) [2]

Cette méthode est basée sur un test d'hypothèses pour établir si la fusion de deux groupes est justifiée ou non.

Considérons un ensemble de n individus décrits par p variables.

Nous définissons tout d'abord

$$W_1 = \sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij} - g_j)^2 = I(C)$$

qui est l'inertie totale d'un ensemble de points, et

$$W_2 = \sum_{l=1}^{2}\sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij}^{(l)} - g_j^{(l)})^2 = I(C_1) + I(C_2)$$

qui représente la somme des inerties de chacune des classes.

W_1 est donc la somme des carrés des distances à l'intérieur d'un ensemble de points et W_2 est la comme des carrés des distances à l'intérieur de deux groupes obtenus en divisant l'ensemble initial en deux.

Nous pouvons établir la statistique du test :

$$W = \frac{\dfrac{W_1 - W_2}{W_2}}{\left(\left(\dfrac{n-1}{n-2}\right) 2^{2/p} - 1\right)} \sim F(p, (n-2)p)$$

où $F(p,(n-2)p)$ représente la loi de Fisher-Snedecor à p degrés de liberté au numérateur et $(n-2)p$ degrés de liberté au dénominateur.

Si k_0 est la première valeur qui conduit au rejet de la fusion de deux groupes, le test de Beale indique que le nombre de classes est $k_0 + 1$.
Pour un niveau de signification $\alpha = 0.05$, $F_{p,(n-2)p} = 5.30$ et pour un niveau de signification $\alpha = 0.01$, $F_{p,(n-2)p} = 4.61$.

Ce test peut toutefois être généralisé. Nous testons alors l'hypothèse nulle sur la présence de k_1 groupes contre l'hypothèse alternative de k_2 groupes. Nous avons alors

$$\frac{\dfrac{W(k_1) - W(k_2)}{W(k_2)}}{\left(\dfrac{(n-k_1)}{(n-k_2)}\left(\dfrac{k_2}{k_1}\right)^{2/p} - 1\right)} \sim F(p(k_2 - k_1), p(n - k_2)).$$

Notons que ce test ne peut pas être appliqué aux méthodes non-hiérarchiques.

5.3 Une méthode basée sur le critère des hypervolumes [10]

Considérons $x = (x_1, \ldots, x_n)$ la réalisation d'un processus de Poisson homogène dans t domaines convexes disjoints dans un espace euclidien à p dimensions.

Nous allons tester si la partition optimale en k classes notée $C = \{C_1, \ldots, C_k\}$ est significativement meilleure que la partition optimale en $(k-1)$ classes $B = \{B_1, \ldots, B_{k-1}\}$.

Le test effectué est le suivant :

$$\begin{cases} H_0 : & \text{les données sont générées dans } k \text{ domaines convexes disjoints} \\ H_1 : & \text{les données sont générées dans } (k-1) \text{ domaines convexes disjoints.} \end{cases}$$

Ce qui équivalent à :

$$\begin{cases} H_0 : & t = k \\ H_1 : & t = k-1. \end{cases}$$

Le quotient de vraisemblance est le suivant :

$$\begin{aligned} Q(x) &= \frac{\dfrac{1}{\Big(\sum_{l=1}^{k-1} m(H(B_l))\Big)^n}}{\dfrac{1}{\Big(\sum_{l=1}^{k} m(H(C_l))\Big)^n}} \\ &= \Big(\frac{W(P^*,k)}{W(P^*,k-1)}\Big)^n \\ &= S^n \end{aligned}$$

Nous rejetterons l'hypothèse nulle H_0 lorsque $Q(x)$ sera "grand", c'est-à-dire lorsque S sera "grand". La région critique aura donc la forme :

$$W = \{S > c\}.$$

Or, la distribution de la statistique S est inconnue, ce qui nous empêche de déterminer le niveau de signification auquel nous rejetterons ou non l'hypothèse nulle. Cependant, nous avons toujours la propriété suivante :

$$W(P^*,k) > W(P^*,k-1).$$

Ceci nous permet d'affirmer que $S \in [0,1[$. Pratiquement, cela nous permet d'utiliser la règle de décision suivante :
rejeter H_0 si S prend de grandes valeurs, c'est-à-dire si S est proche de 1.

Si k_0 est la première valeur pour laquelle nous rejetons H_0, nous considérerons $k_0 - 1$ comme le nombre de classes naturelles présentes dans les données.

Chapitre 6

Description du programme Sclust

6.1 Introduction

Sclust est une méthode symbolique de classification en cours de développement. Elle est réalisée dans le cadre du contrat européen ASSO[1]. Ce programme, conçu et réalisé par Yves Lechevallier de l'INRIA Rocquencourt, est écrit en C++.

Sclust est basé sur la méthode des nuées dynamiques classiques. A chaque itération, ce programme détermine une partition de l'ensemble des objets de façon à améliorer un critère de classification. Il calcule à chaque étape les prototypes qui représentent les classes et évalue le critère en fonction d'une mesure de proximité qui détermine la ressemblance entre les objets d'une classe et leur représentant.

Sclust traite les variables classiques ainsi que les variables de type intervalle, les variables multivaluées et les variables modales.

6.2 Le fichier Sodas

Ce fichier contient tous les renseignements que Sclust nécessite sur le jeu de données pour pouvoir effectuer la classification. Le fichier Sodas est généré à l'aide du module `DB2SO` du logiciel Sodas qui transforme une base de données Excel ou Access dans le format adéquat pour Sclust.

[1] Analysis System of Symbolic Official data

Le format du fichier Sodas comprend au minimum les six blocs suivants :
CONTAINS reprend la liste de tous les blocs contenus dans le fichier Sodas.
FILE donne des informations sur l'origine du fichier.
HEADER renseigne

- le titre et le sous-titre du jeu de données,
- le nombre total d'individus et de variables dans le fichier de données,
- le nombre de variables de chaque type,
- le nombre de données manquantes.

INDIVIDUALS assigne à chaque individu un numéro et un libellé.
VARIABLES affecte à chaque variable un numéro, son type, un libellé et le nombre de modalités qu'elle peut prendre. Ensuite, à chaque modalité est associé un numéro et un libellé.
RECTANGLE_MATRIX correspond à la matrice de données symboliques. La $i^{\text{ème}}$ ligne de la matrice représente la description du $i^{\text{ème}}$ individu, alors que la $j^{\text{ème}}$ colonne donne les valeurs prises par la $j^{\text{ème}}$ variable pour tous les individus.

En plus de ces six blocs, nous pouvons avoir :
RULES qui spécifie la restriction des valeurs que peut prendre une certaine variable Y_k si celle-ci est dépendante d'une autre variable Y_l.
TRIANGLE_MATRIX qui est la matrice triangulaire inférieure des dissimilarités entre tous les individus du jeu de données.

Exemple 6.2.1

Nous avons créé un fichier Sodas de données artificielles. Ce jeu de données est composé de 12 objets décrits par une variable modale et une variable multivaluée. Notre fichier contient uniquement les six blocs de base.

```
SODAS = (
CONTAINS = (
   FILES, HEADER, INDIVIDUALS, VARIABLES, RECTANGLE_MATRIX
),
FILE =  (
   procedure_name = "db2so" ,
   version = "sans" ,
   create_date = ""
),
HEADER =  (
   title = "Données artificielles" ,
   sub_title = "Données artificielles" ,
   indiv_nb = 12 ,
   var_nb = 2 ,
   rules_nb = 0 ,
   nb_var_set = 0 ,
   nb_indiv_set = 0 ,
   nb_var_nom = 0 ,
   nb_var_cont = 0 ,
```

```
   nb_var_text = 0 ,
   nb_var_cont_symb = 0 ,
   nb_var_nom_symb = 1 ,
   nb_var_nom_mod = 1 ,
   nb_na = 0 ,
   nb_null = 0 ,
   nb_nu = 0 ,
   nb_hierarchies = 0
),

INDIVIDUALS = (
  (0,"i0", "obj1" ),
  (1,"i1", "obj2" ),
  (2,"i2", "obj3" ),
  (3,"i3", "obj4" ),
  (4,"i4", "obj5" ),
  (5,"i5", "obj6" ),
  (6,"i6", "obj7" ),
  (7,"i7", "obj8" ),
  (8,"i8", "obj9" ),
  (9,"i9", "obj10" ),
  (10,"i10", "obj11" ),
  (11,"i11", "obj12" )
 ),
VARIABLES =  (
 (1 ,mult_nominal_Modif ,"" ,"v1" ,"nom var modale" ,0, 0 ,3, (
        (1 ,"v1/1" ,"a" ,0),
        (2 ,"v1/2" ,"b" ,0),
        (3 ,"v1/3" ,"c" ,0 ) )
  ),
 (2 ,mult_nominal ,"" ,"v2" ,"nom var multivaluee" ,0, 0 ,2, (
        (1 ,"v2/1" ,"x" ,0),
        (2 ,"v2/2" ,"y" ,0 ) )
  )
),
RECTANGLE_MATRIX = (
 ((1(1)), (1,2)),
 ((1(0.5),2(0.5)), (1,2)),
 ((1(1)), (2(1))),
 ((1(0.5),3(0.5)), (1,2)),
 ((2(1)), (1(1))),
 ((3(1)), (2(1))),
 ((1(0.333333),2(0.333333),3(0.333333)), (1,2)),
 ((1(0.5),2(0.5)), (1,2)),
 ((1(0.5),3(0.5)), (2)),
 ((2(0.5),3(0.5)), (1)),
 ((1(1)), (2)),
 ((2(0.333333),3(0.666667)), (1))
))
 END
```

6.3 Le fichier Scluster.h

Le fichier Scluster.h contient une structure appelée `scluster`. En C++, les structures permettent de regrouper des objets (des variables) au sein d'une entité repérée par un seul nom de variable. La structure présente dans ce fichier permet de stocker différents types de données, en particulier :

- le nombre d'objets dans le fichier sodas ;
- le nombre de variables dans le fichier sodas ;
- le nombre d'objets sélectionnés ;
- le nombre de variables sélectionnées ;
- le nombre de classes recherchées ;
- le nombre d'itérations fixé par l'utilisateur ;
- le nombre d'essais demandé par l'utilisateur ;
- le code de la distance choisie par l'utilisateur ;
- le code de l'initialisation choisie par l'utilisateur ;
- etc.

Il est évident que les valeurs prises par chacune de ces variables dépendent du fichier sodas sélectionné ainsi que des choix faits par l'utilisateur et ne changent pas tout au long de l'exécution du programme.

Regardons plus en détails comment est définie la structure `scluster`.

```
struct scluster {
        char listing [maxlenfile+2]; // nom du fichier listing
        char file_log[maxlenfile+2]; // nom du fichier log
        char clustersds[maxlenfile +2];// nom du fichier sds contenant
                                       // les classes
        char prototypesds[maxlenfile +2];// nom du fichier sds
                                         // contenant les prototypes
        char nombase[maxlenfile+2]; // nom du fichier de sortie sds
        char file_tex[maxlenfile+2];// nom du fichier de sortie latex
        long nb_objects_base; // nombre d'objets dans le fichier sodas
        long nb_variables_base;// nombre de variables dans le fichier sodas
        long nb_objects;// nombre d'objets sélectionnés
        long nb_variables;// nombre de variables sélectionnées
        int nb_class;// nombre de classes
        int nb_iter;// nombre d'itérations
        int nb_run;// nombre d'essais
        int distance_quant;// code distance var. quantitatives ou intervalles
        int distance_qual;// code distance var. qualitatives
        int distance_modal;// code distance var. multivaluées ou modales
        int normalize; // code de la normalisation
        int init;// code de l'initialisation
        int nb_mod;// nombre total de modalités
        int maxlibvar;// longueur maximum du libellé des variables
```

```
int maxlibobj;// longueur maximum du libellé des objets
int nb_col;// nombre de colonnes de la liste des objets
int debug;// code du debug
};
```

Exemple 6.3.1
Reprenons le fichier sodas de l'exemple 6.2.1. Supposons que l'utilisateur souhaite effectuer 5 essais de recherche de la partition optimale en 3 classes des 12 objets décrits par la variable modale (qui peut prendre trois modalités), chaque essai comprenant au plus 10 itérations de la méthode des nuées dynamiques, qu'il ait choisi la distance de De Carvalho et qu'il veuille une initialisation par une partition aléatoire. Alors, les différentes variables de la structure prendront les valeurs suivantes :

- `nb_objects` = 12
- `nb_variables` = 1
- `nb_class` = 3
- `nb_iter` = 10
- `nb_run` = 5
- `distance_modal` = 0
- `init` = 0
- `nb_mod` = 3

De plus, le fichier `Scluster.h` contient les déclarations des fonctions des fichiers `calcul_scluster.cpp` et `edit_scluster.cpp`.

6.4 Le fichier calcul_scluster.cpp

Le fichier `calcul_scluster.cpp` est le fichier le plus important de Sclust. Il contient quatre fonctions qui correspondent aux quatre principales étapes de la méthode des nuées dynamiques dans le cas de données symboliques. Nous allons décrire ces quatre fonctions ainsi que celle qui contient l'algorithme des nuées dynamiques.

6.4.1 La fonction Init_scluster

La fonction `Init_scluster` permet d'initialiser la partition en choisissant soit une initialisation par une partition aléatoire ou soit une initialisation par prototypes.

Initialisation par une partition aléatoire

Dans ce cas-ci, la fonction `Init_scluster` va affecter chaque objet à une classe de façon aléatoire. Pour ce faire, elle va procéder comme suit :

```
// Initialisation par une partition aleatoire
for (i=1;i<=imax;i++) // Boucle sur les individus
  {
    x1=float(rand())/float(RAND_MAX);
    x1=x1*(float)kmax;
    k=int (x1)+1;classe[i]=k;
    Nt[k]=Nt[k]+1;
  };

// Construction des prototypes
Proto_scluster(lis,para,M,vsel,classe,Proto);

// Calcul du critère
ww=W_scluster(lis,para,M,vsel,classe,Pw,Proto);
```

Pour chaque individu contenu dans le jeu de données, la fonction `Init_scluster` associe un nombre aléatoire réel strictement compris entre 0 et `nb_class`, le nombre de classes demandé par l'utilisateur. Ensuite, ce nombre est tronqué à l'entier supérieur et c'est ce nombre qui détermine la classe initiale d'appartenance de l'individu considéré.

La fonction compte aussi le nombre d'individus présents dans chaque classe. Une fois que chaque individu est assigné à une classe, la fonction détermine le représentant initial de chaque classe en appelant la fonction `Proto_scluster` ainsi que la valeur du critère à l'aide de la fonction `W_scluster`.

Initialisation par prototypes

La fonction associe aléatoirement à chaque classe un nombre aléatoire réel strictement compris entre 0 et `nb_objects`. Ce nombre est ensuite tronqué à l'entier supérieur. Il détermine alors l'individu qui servira de prototype de la classe considérée.

Remarquons aussi que deux classes distinctes ne peuvent pas être représentées par le même prototype.

6.4.2 La fonction affec_scluster

La fonction `affec_scluster` correspond à la fonction d'affectation f de la méthode des nuées dynamiques.

Cette fonction recherche le prototype duquel chaque individu est le plus proche suivant la distance choisie par l'utilisateur et affecte l'individu considéré à la classe représentée par ce prototype.

Regardons comment cette fonction procède pour des variables multivaluées catégoriques ainsi que pour des variables modales.

```
for (i=1;i<=imax;i++) // Pour chaque individu
{
  for (k=1;k<=kmax;k++) // Pour chaque classe
  {
    dd=0.;l=0;
    for (j=0;j<nvar;j++) // Pour chaque variable
    {
      num=vsel[j]; // num est le numéro de la variable sélectionnée en cours
      (...)
      switch(M->gettypevar(num)) // Type de la variable sélectionnée
      {
        (...)
        case multnomin: // Cas des variables multivaluées
        {
          nmod=M->getnbcateg(num);
          total=((dim_Mat*)M)->nomicard(i,num);
          for (kk=1;kk<=nmod;kk++)
          {
            l++;
            if(total == 0)
              xx=M->index(i,num).getnomin()->index(kk);
            else
              xx=((double)M->index(i,num).getnomin()->index(kk))/total;
            switch(para.distance_modal) // Type de la distance choisie
            {
            case 0: // De Carvalho
              if (xx==0.) dd=dd+Proto[k][l];
              if (Proto[k][l]==0.) dd=dd+xx;
              break;
            case 1: // L1
              dd=dd+fabs(Proto[k][l] - xx);
              break;
            case 2: // L2
              dd=dd+(Proto[k][l] - xx)*(Proto[k][l] - xx);
              break;
            default :
              if ((int)100.0*xx==0.) dd=dd+Proto[k][l];
```

```
          if ((int)100.0*Proto[k][l]==0.) dd=dd+xx;
        };
      };
    };
    break;
    case multnominm: // Cas des variables modales
    {
      nmod=M->getnbcateg(num);
      total=((dim_Mat*)M)->fuzzycard(i,num);
      for (kk=1;kk<=nmod;kk++)
      {
        l++;
        if(total == 0)
          xx=M->index(i,num).getnominM()->indexm(kk);
        else
          {
          xx=M->index(i,num).getnominM()->indexm(kk)/total;
          };
        switch(para.distance_modal)
        {
        case 0: // De Carvalho
          if ((int)100.0*xx==0.) dd=dd+Proto[k][l];
          if ((int)100.0*Proto[k][l]==0.) dd=dd+xx;
          break;
        case 1: // L1
          dd=dd+fabs(Proto[k][l] - xx);
          break;
        case 2: // L2
          dd=dd+(Proto[k][l] - xx)*(Proto[k][l] - xx);
          break;
        default :
          if ((int)100.0*xx==0.) dd=dd+Proto[k][l];
          if ((int)100.0*Proto[k][l]==0.) dd=dd+xx;
        };
      };
    };
    break;
    (...)
  };
};// Fin de la boucle sur le nombre de variables
if(k==1)
{
  dmin=dd;nclass=1;
}
else
{
  if(dmin > dd)
  {
    dmin=dd;nclass=k;
  };
```

```
    };
  }; // Fin de la boucle sur le nombre de classes
  if(nclass != classe[i]) n_diff++;
  classe[i]=nclass;ww=ww+dmin;
}; // Fin de la boucle sur le nombre d'individus
return n_diff;
```

Remarquons d'abord que la totalité de l'algorithme n'a pas été présentée ici. Les parties manquantes ont été marquées par `(...)`.

Nous allons d'abord décrire la boucle la plus interne de cet algorithme, c'est-à-dire la boucle sur le nombre de variables. Supposons fixés l'individu i et la classe k.

D'après le type des variables sélectionnées, l'algorithme va passer dans un des deux cas suivants :

1. **Le cas des variables multivaluées**
   La fonction calcule d'abord `nmod` et `total` qui contiennent respectivement le nombre de catégories que peut prendre la variable `num` sélectionnée et le nombre de catégories que prend effectivement la variable `num` pour l'individu i.
   Ensuite, pour chaque catégorie, l'algorithme calcule `xx` qui vaut :
   $$\texttt{xx} = \begin{cases} \texttt{1/total} & \text{si la catégorie considérée correspond à une des catégories} \\ & \text{prises par la variable num de l'individu } i \\ 0 & \text{sinon.} \end{cases}$$
   `xx` représente donc la valeur de la distribution associée à la catégorie `kk` de la variable `num`.
   Trois cas se présentent alors en fonction de la distance choisie par l'utilisateur :
   - la distance de De Carvalho, `(case 0)` ;
   - la distance absolue (L_1), `(case 1)` ;
   - la distance euclidienne (L_2), `(case 2)`.

   L'algorithme passe dans le `case` correspondant et met à jour la distance `dd` entre l'individu i et le prototype de la classe k. Cette mise à jour consiste en l'ajout, pour chaque catégorie que peut prendre la variable `num`, de la distance entre le prototype considéré et `xx` à la somme des distances déjà calculées pour les variables précédentes.
2. **Le cas des variables modales**
   Le raisonnement pour les variables modales est fort semblable à celui décrit ci-dessus pour les variables multivaluées.
   L'algorithme commence par calculer `nmod` qui contient le nombre de modalités que peut prendre la variable `num` sélectionnée.

Nous avons vu dans le chapitre 2 (section 2.4.1) qu'une variable modale est décrite par $Y(x_i) = (U(x_i), \pi_i)$, $\forall x_i \in E$ où π_i est une mesure ou une distribution (poids, probabilité ou fréquence).
La variable `total` représente la somme des valeurs prises par cette mesure pour toutes les modalités de la variable `num` pour l'individu i. Notons que dans le cas où π_i est une distribution de probablilité ou de fréquence, cette somme vaudra 1.

Ensuite, pour chaque modalité, l'algorithme calcule `xx` qui vaut

$$\texttt{xx} = \begin{cases} \pi_i/\texttt{total} & \text{si la modalité considérée correspond à une des} \\ & \text{modalités prises par la variable num de l'individu } i \\ 0 & \text{sinon.} \end{cases}$$

En d'autres mots, `xx` correspond donc à une normalisation de la mesure ou distribution associée à la modalité `kk` de la variable `num`.

Les trois distances utilisées sont les mêmes que décrites ci-dessus pour les variables multivaluées.

Une fois que toutes les modalités pour toutes les variables ont été prises en compte, l'algorithme compare la distance totale entre l'individu i et la classe k avec la distance minimale obtenue entre ce même individu i et une des $k-1$ classes considérées précédemment.

Si la distance est minimale, la distance `dmin` et la classe d'affectation de l'individu i sont modifiées.

La fonction `affec_scluster` calcule enfin le nombre de changements de classe d'affectation `n_diff` de tous les individus.

6.4.3 La fonction W_scluster

Cette fonction permet de calculer la valeur du critère de la méthode des nuées dynamiques. Ce critère mesure l'adéquation entre toute partition et toute représentation de cette partition et vaut :

$$W(P, L) = \sum_{l=1}^{k} \sum_{x_i \in C_l} d(x_i, L_l)$$

où
- k est le nombre de classes ;
- C_l est la classe l ;
- L_l est le représentant de la classe l.

Voici les principales étapes de l'algorithme.

```
for (i=1;i<=imax;i++) // Boucle sur les individus
{
   dd=0.; // Initialisation de la distance à 0
   (...)
   k=classe[i]; // k est l'indice de la classe d'appartenance de i
   for (j=0;j<nvar;j++) // Boucle sur les variables
   {
      num=vsel[j]; // numéro de la variable sélectionnée
      (...)
      switch(M->gettypevar(num))
        {
        (...)
        case multnomin: // Cas des variables multivaluées
                {
                nmod=M->getnbcateg(num);
                total=((dim_Mat*)M)->nomicard(i,num);
                for (kk=1;kk<=nmod;kk++)
                 {
                 l++;
                 if(total == 0)
                   xx=M->index(i,num).getnomin()->index(kk);
                 else
                   xx=((double)M->index(i,num).getnomin()->index(kk))/total;
                 switch(para.distance_modal) // Type de la distance choisie
                 {
                 case 0: // De Carvalho
                   if (xx==0.) dd=dd+Proto[k][l];
                   if (Proto[k][l]==0.) dd=dd+xx;
                   break;
                 case 1: // L1
                   dd=dd+fabs(Proto[k][l] - xx);
                   break;
                 case 2: // L2
                   dd=dd+(Proto[k][l] - xx)*(Proto[k][l] - xx);
                   break;
                 default :
                   if ((int)100.0*xx==0.) dd=dd+Proto[k][l];
                   if ((int)100.0*Proto[k][l]==0.) dd=dd+xx;
                 };
                 };
                };
                break;
                case multnominm: // Cas des variables modales
                {
                nmod=M->getnbcateg(num);
                total=((dim_Mat*)M)->fuzzycard(i,num);
                for (kk=1;kk<=nmod;kk++)
                 {
                 l++;
```

```
                    if(total == 0)
                       xx=M->index(i,num).getnominM()->indexm(kk);
                    else
                     {
                       xx=M->index(i,num).getnominM()->indexm(kk)/total;
                     };
                    switch(para.distance_modal) // Type de la distance choisie
                    {
                    case 0: // De Carvalho
                      if ((int)100.0*xx==0.) dd=dd+Proto[k][l];
                      if ((int)100.0*Proto[k][l]==0.) dd=dd+xx;
                      break;
                    case 1: // L1
                      dd=dd+fabs(Proto[k][l] - xx);
                      break;
                    case 2: // L2
                      dd=dd+(Proto[k][l] - xx)*(Proto[k][l] - xx);
                      break;
                    default :
                      if ((int)100.0*xx==0.) dd=dd+Proto[k][l];
                      if ((int)100.0*Proto[k][l]==0.) dd=dd+xx;
                    };
                    };
                    (...)
                   };
          };
     }; // Fin de la boucle sur les variables
ww=ww+dd; // Mise à jour de la valeur du critère
num=classe[i]; // num est le numéro de la classe de l'individu i
Wt[num]=Wt[num]+dd; // Wt[num] est l'inertie intra-classes de la
                    // classe num
}; // Fin de la boucle sur les individus
return ww;
```

Comme la fonction `affec_scluster`, la fonction `W_scluster` détermine la distance entre chaque individu et le prototype de sa classe d'affectation. Ce qui diffère avec la fonction `affec_scluster`, c'est qu'ici nous connaissons la classe d'affectation de l'individu i et qu'il n'y a donc pas de boucle sur le nombre de classes.

La fonction renvoie au programme principal la valeur de la variable `ww` qui représente la valeur du critère de la méthode des nuées dynamiques. C'est donc la valeur de `ww` que nous cherchons à minimiser. `ww` est la somme des écarts entre les individus de chaque classe et leur représentant, c'est-à-dire l'inertie intra-classes W du nuage de points.

De plus, la fonction `W_scluster` calcule la contribution de chaque classe à l'inertie intra-classes du nuage de points et stocke ces valeurs dans le vecteur `Wt[ ]`. Ceci se calcule facilement grâce au fait que l'inertie intra-classes d'un nuage de points peut se décomposer de manière additive sur les classes.

6.4.4 La fonction Proto_scluster

Une fois que chaque individu a été affecté à la classe dont il est le plus proche, il faut reconstruire le prototype de chaque classe.

La fonction `Proto_scluster` correspond à la fonction de représentation g de la méthode des nuées dynamiques. Elle se charge de déterminer, sur base des individus présents dans chaque classe, le représentant de celle-ci. Ces prototypes sont ensuite stockés dans la matrice `Proto[ ][ ]`, où `Proto[k][l]` contient le prototype de la classe k relatif à la modalité l.

Regardons comment sont calculés les prototypes dans le cas de variables multivaluées et modales.

Supposons une variable `num` et un individu i fixés. Notons que `nclas` représente le numéro de la classe à laquelle l'individu i est affecté.

```
l=1;
for (j=0;j<nvar;j++) // Boucle sur les variables
{
 num=vsel[j];
 for (i = 1;i<=imax;i++) // Boucle sur les individus
  {
    nclas=classe[i];
    if ( M->index(i,num).na() || M->index(i,num).nu() )
        (...)
    else
        {
        switch(M->gettypevar(num))
        {
          case multnomin: // Cas des variables multivaluées
          {
            nmod=M->getnbcateg(num);
            total=((dim_Mat*)M)->nomicard(i,num);
            l1=l-1;
            if((int)total == 0)
              {
                for (k=1;k<=nmod;k++)
                  {
                    l1++;
                    Proto[nclas][l1]= Proto[nclas][l1] +
```

```
                    M->index(i,num).getnomin()->index(k);
                };
            }
          else
            {
              Pt[nclas][j]=Pt[nclas][j]+1;
              for (k=1;k<=nmod;k++)
                {
                  l1++;
                  Proto[nclas][l1]= Proto[nclas][l1] +
                       (M->index(i,num).getnomin())->index(k)/total;
                };
            };
        };
        break;
        case multnominm: // Cas des variables modales
        {
          nmod=M->getnbcateg(num);
          total=((dim_Mat*)M)->fuzzycard(i,num);l1=l-1;
          if(total == 0)
            {
              for (k=1;k<=nmod;k++)
                {
                  l1++;
                  Proto[nclas][l1]=Proto[nclas][l1] +
                       M->index(i,num).getnominM()->indexm(k);
                };
            }
          else
            {
              Pt[nclas][j]=Pt[nclas][j]+1;
              for (k=1;k<=nmod;k++)
                {
                  l1++;
                  Proto[nclas][l1]=Proto[nclas][l1] +
                       M->index(i,num).getnominM()->indexm(k)/total;
                };
            };
        };
        break;
      };
    };
  };// Fin de la boucle sur les individus
}; // Fin de la boucle sur les variables
```

La fonction met d'abord à jour la matrice `Pt[ ]` qui contient, pour chacune des `nclas` classes, le nombre d'individus qui la composent. Elle met ensuite à jour la matrice des prototypes `Proto[ ][ ]`.

Chaque colonne de la matrice `Proto[ ][ ]` correspond à une des modalités que peut prendre l'individu i pour chaque variable.

Après avoir analysé tous les individus, `Proto[nclas][l1]` contient la somme des distributions associées à chaque catégorie que prend la variable j pour décrire les individus appartenant à la classe `nclas`.

Il reste encore à normaliser les éléments de la matrice, c'est-à-dire à diviser chaque ligne de la matrice, correspondant chacune à une classe, par le nombre d'individus de cette classe pour que la matrice `Proto` contienne alors les prototypes de chaque classe.

```
for (i=1;i<=nmod;i++)
  {
    l1++;
    for (k=1;k<=kmax;k++) // Boucle sur les classes
      {
        if(Pt[k][j] != 0)
        Proto[k][l1]=Proto[k][l1]/(double)Pt[k][j];
      };
  };
```

6.4.5 La fonction main_scluster

Cette fonction va appeler les différentes fonctions vues précédemment de manière à constituer l'algorithme des nuées dynamiques.

Supposons que la structure `scluster` est initialisée et regardons comment cet algorithme fonctionnne.

```
for (n=1;n<=n_run;n++) // Boucle sur le nombre de runs
{
  // Initialisation de la méthode
  Init_scluster(lis,para,M,vsel,classe, Nt, Proto,Pw);
  iter=0;
  n_diff=imax;
  while( (iter < iter_max) && (n_diff != 0)) // Boucle sur les itérations
  {
    iter=iter+1;

    /************* Etape d'affectation *************/
    // Calcul de la partition
    n_diff=Affec_scluster(lis,para,M,vsel,classe,Proto,Pw);
    // Calcul du critère
    ww=W_scluster(lis,para,M,vsel,classe,Pw,Proto);
```

```
  /************** Etape de représentation *********/
  // Calcul des prototypes
  Proto_scluster(lis,para,M,vsel,classe,Proto);
  // Calcul du critère
  ww=W_scluster(lis,para,M,vsel,classe,Pw,Proto);
}; // Fin de la boucle sur les iterations

// stockage des informations sur ce run
W_opt[n]=ww;N_iter[n]=iter;Nb_classe[n]=0;
for (k=1;k<=kmax;k++) Nt[k]=0;
for (i=1;i<=imax;i++) Nt[classe[i]]++;
for (k=1;k<=kmax;k++) if(Nt[k] != 0) Nb_classe[n]++;

/***** Si solution optimale : stockage du run ******/
if(n==1)
  {
    max_run=1;w_max=ww;
    for (i=1;i<=imax;i++)
    classe_opt[i]=classe[i];
    for (k=1;k<=kmax;k++)
      {
      for (m=0;m<=mod_max;m++)
        {
          Proto_opt[k][m]=Proto[k][m];
        };
      };
  }
else
  {
    if(ww < w_max)
      {
        max_run=n;w_max=ww;
        for (i=1;i<=imax;i++)
              classe_opt[i]=classe[i];
        for (k=1;k<=kmax;k++)
          {
              for (m=0;m<=mod_max;m++)
                {
                      Proto_opt[k][m]=Proto[k][m];
                };
          };
      };
  };
}; // Fin de la boucle sur le nombre de runs
```

Supposons fixé un essai (run) de recherche de la partition optimale en k classes. Notons cet essai n.

L'algorithme calcule tout d'abord, grâce à la fonction `Init_scluster`, la partition initiale et le prototype de chacune des classes.

Ensuite, à chaque itération, la fonction va appeler
- la fonction `affec_scluster` qui va affecter chaque individu au prototype dont il est le plus proche,
- la fonction `proto_scluster` qui va calculer le nouveau prototype de chaque classe,
- la fonction `W_scluster` qui va déterminer la valeur du critère.

Remarquons que l'algorithme se termine soit quand le nombre d'itérations maximal défini par l'utilisateur est atteint ou soit quand il n'y a plus aucun changement `n_diff` de classe d'affectation des individus.

Cette dernière condition est due au fait que la suite $v_i = (P_i, L_i)$, composée de la partition et de ses représentants à l'itération i, devient stationnaire après un certain nombre N d'itérations, i.e.

$$\exists N \text{ tel que } \forall i > N,\ v_i = v_{i+1}.$$

L'algorithme fait donc décroître la valeur du critère jusqu'à stabilisation et il n'y a alors plus aucun intérêt à continuer les itérations.

Ensuite, plusieurs informations concernant l'essai en cours sont stockées :
- `W_opt[n]` contient la valeur optimale du critère,
- `N_iter[n]` contient le nombre d'itérations effectuées pour obtenir la solution,
- `Nt[classe[i]]` contient le nombre d'individus par classe,
- `Nb_classe[n]` contient le nombre de classes.

Pour terminer, si la valeur du critère calculée par l'essai considéré n est inférieure à celles des $n-1$ essais effectués précédemment, alors la fonction stocke cet essai ainsi que :
- la valeur du critère `ww` dans `w_max`,
- la partition optimale dans `classe_opt[i]`,
- les prototypes de chaque classe dans `Proto_otp[k][m]`.

6.4.6 Le fichier listing

Le fichier listing est le fichier dans lequel les résultats fournis par Sclust sont sauvegardés.

Décrivons les sorties obtenues en appliquant Sclust sur le jeu de données artificielles présenté précédemment.

Premièrement, Sclust rappelle le nom du fichier sodas contenant le jeu de données ainsi que les noms des différents fichiers de sortie qu'il a créés.

```
Sodas File            :     C:\slcust\artif.sds
Log   File            :     C:\sclust\artif.log
Listing   File        :     C:\sclust\artif.lst
Latex File            :     C:\sclust\artifRES.tex
Sodas Out Cluster     :     C:\sclust\artifSclustCluster.sds
Sodas Out Prototype   :     C:\sclust\artifSclustPrototype.sds
```

Le fichier `artifSclustCluster.sds` décrit pour chaque partition la classe d'appartenance de chaque individu. Le fichier `artifSclustPrototype.sds` donne pour chaque partition les prototypes de chaque classe. Enfin, le fichier `artifRES.tex` fournit un résumé des résultats obtenus pour chaque partition.

Dans un second temps, Sclust affiche les valeurs prises par les principales variables de la structure `scluster`. Toutes ces valeurs restent inchangées tout au long de l'exécution du programme. Il indique également la valeur du critère initial, c'est-à-dire la valeur de l'inertie totale.

```
Learning Set          :    12
Number of variables   :     2
Number of iterations  :    10
Number of classes     :     3
Initialisation        :     0  random partition
Number of runs        :     2
Quantitave distance   :     0  Hausdorff Distance
Qualitative distance  :     0  De Carvalho Distance
Modal distance        :     0  De Carvalho Distance
Normalize             :     0  No
Number of total modalities :     5

Initial Criterion : 8.583333
```

Ensuite, pour chaque variable sélectionnée est spécifié
- son numéro ;
- son nom ;
- son type ainsi que le nombre de modalités ;

- les valeurs $100.\frac{T_j}{T}$ et $100.\frac{W_j}{W}$, c'est-à-dire les contributions de chaque variable à l'inertie totale et à l'inertie intra-classes. Etant donné qu'aucune classification n'a encore été effectuée jusqu'à présent, l'ensemble des individus forme une seule classe et ces deux valeurs sont égales.
- son poids.

```
GROUP OF SELECTED VARIABLES :
==============================
 ( Pos )  Name          Type                  Initial Bj  Bj used  Weight

 (   1 ) First      MULTINOMINAL 3 Modalities    59.55    59.55   1.000000
 (   2 ) Second     MODAL        2 Modalities    40.45    40.45   1.000000
```

Il continue en donnant la liste des objets symboliques contenus dans le jeu de données.

```
LIST OF SYMBOLIC OBJECTS IN THE SET :
=====================================
 "obj1"      "obj2"      "obj3"      "obj4"      "obj5"      "obj6"
 "obj7"      "obj8"      "obj9"      "obj10"     "obj11"     "obj12"
```

Ensuite, pour chaque essai, Sclust décrit chaque itération en indiquant le nombre de changements de classe d'affectation de tous les individus et la valeur du critère. Remarquons que la valeur du critère est strictement décroissante.

```
RUN NUMBER :    1
================
  Iteration Permutation Criterion
      1          4       5.571429
      2          0       4.351852

RUN NUMBER :    2
================
  Iteration Permutation Criterion
      1          5       6.722222
      2          0       4.500000
```

A la fin de tous les essais, Sclust fournit la solution optimale qui correspond à l'essai pour lequel la valeur du critère est minimale ainsi que la valeur de l'inertie intra-classes.

```
OPTIMAL SOLUTION
================
  RUN NUMBER :  1
  CRITERION  :  4.351852
```

Sclust décrit alors la partition optimale en donnant pour chaque classe le nombre d'individus qu'elle contient ainsi que la liste des objets qui la composent. De plus, la distance de chaque objet au prototype de sa classe est mentionnée.

```
EDITION OPTIMAL PARTITION
=========================
 Classe :   1  Cardinal :      9
 ================================
 (  0) "obj1"     [0.9]  (  5) "obj3"     [2.4]  (  6) "obj4"  [0.0]
 (  7) "obj5"     [0.6]  (  8) "obj6"     [1.0]  ( 10) "obj8"  [1.6]
 (  1) "obj10"    [0.6]  (  2) "obj11"    [1.6]  (  3) "obj12" [0.3]

 Classe :   2  Cardinal :      3
 ================================
  (  4) "obj2"    [3.0]  (  9) "obj7"     [0.0]  ( 11) "obj9"  [0.0]
```

Après cela, nous avons la description de la partition pour laquelle sont calculées les valeurs suivantes :

- la dispersion du nuage de points qui équivaut à l'inertie totale T du nuage de points,
- la valeur du critère qui est égal à l'inertie intra-classes W,
- le pourcentage de dispersion qui vaut :

$$100.\frac{T-W}{T} = 100.\frac{B}{T}.$$

```
PARTITION DESCRIPTION
=====================
 DISPERSION : 8.583333
 CRITERION  : 4.351852
 Pourcentage of dispersion :    49.30
```

Vient alors la description des variables. Pour chacune d'entre elles, les indices suivants sont calculés :

- $100.\frac{B_j}{T_j} = cor(j)$ qui représente la part d'inertie de la variable j pris en compte par la partition,
- $100.\frac{W_j}{W}$ qui mesure la contribution relative de chaque variable à l'inertie intra-classes,
- $100.\frac{T_j}{T}$ qui mesure la contribution relative de chaque variable à l'inertie totale,
- Q_2 qui vaut, si nous notons P le pourcentage de dispersion,

$$100.\left(\frac{100.\frac{B_j}{T_j}}{P} - 1\right).$$

```
VARIABLES DESCRIPTION
=====================
 Position        Name       Bj/Tj      Wj/W      Tj/T        Q2
 (   1 )         First      38.04     45.95     59.55    -22.83
 (   2 )         Second     65.87     54.05     40.45     33.61
```

Sclust continue en donnant une description des classes. Pour cela, il commence par donner le nombre d'individus présents dans chacune d'entre elles ainsi que les indices :

- $100.\frac{B_k}{T_k}$,
- $100.\frac{W_k}{W}$ qui représente la contribution relative de chaque classe à l'inertie intra-classes,
- $100.\frac{T_k}{T}$ qui représente la contribution relative de chaque classe à l'inertie totale,
- $100.\frac{B_k}{N_k.B}$,
- $100.\frac{W_k}{N_k.W}$,

où

- B_k est l'inertie inter-classes de la classe k,
- W_k est l'inertie intra-classes de la classe k,
- T_k est l'inertie totale de la classe k,
- N_k est le nombre d'individus dans la classe k.

```
CLUSTER DESCRIPTION
===================
  Cluster  Size(Nk)  Bk/Tk       Wk/W       Tk/T     Bk/Nk.B    Wk/Nk.W
     1         9     26.27      91.06      62.62      10.118      6.958
     2         3     87.88       8.94      37.38       2.979     12.460
```

L'édition des prototypes variable par variable nous indique le prototype de chaque classe en donnant pour chaque modalité la probabilité prise par le représentant de chacune des classes.

```
EDITION PROTOTYPES BY VARIABLES
===============================
 Variable ( 1 ) First
     1         2        <= Cluster
    0.59      0.00     a
    0.15      0.61     b
    0.26      0.39     c

 Variable ( 2 ) Second
     1         2        <= Cluster
    0.30      1.00     x
    0.70      0.00     y
```

Pour terminer, Sclust résume pour chaque essai :

- le nombre d'itérations effectuées,
- le nombre de classes trouvées,
- la valeur optimale du critère,

et se base sur ces valeurs pour donner des statistiques sur la distribution du critère, c'est-à-dire :

- la valeur minimale du critère,
- la moyenne de tous les critères,
- la valeur maximale du critère,
- la déviation standard.

```
CRITERION
=========
  Run    Iteration Class    Criterion
   1      2          2       4.351852
   2      2          2       4.500000

Statistics on criterion distribution :
  Minimum             4.351852
  Means               4.425926
  Maximum             4.500000
  Standard deviation  0.074074
```

Chapitre 7

Adaptation des méthodes de détermination du nombre de classes aux données symboliques

7.1 Introduction

L'objectif principal de ce mémoire est d'adapter les méthodes de détermination du nombre de classes de Milligan et Cooper au programme Sclust en ce qui concerne les objets décrits par des variables multivaluées catégoriques et modales.

L'utilisation de ces variables, contrairement aux variables classiques, nous conduit à trouver un nouveau mode de représentation des objets symboliques. De plus, les méthodes de détermination du nombre de classes sont définies initialement pour des données quantitatives classiques, ce qui nous amène à devoir convertir ces objets en données classiques.

Nous verrons dans ce chapitre les modifications et adaptations effectuées pour insérer les méthodes de détermination du nombre de classes au programme Sclust en ce qui concerne les variables multivaluées catégoriques et modales.

7.2 Visualisation des objets symboliques - La représentation Zoom Star [3]

Nous avons choisi de représenter graphiquement les objets symboliques décrits par des variables multivaluées et modales en utilisant l'éditeur d'objets symboliques (SOE) du logiciel Sodas. Cet éditeur permet de visualiser la représentation graphique d'un objet symbolique sous forme de Zoom Star. Ce Zoom Star peut être vu en deux ou trois dimensions, chaque axe correspondant à une variable. Les représentations en deux et trois dimensions ont chacune leurs particularités. C'est pourquoi nous allons les expliquer séparément.

7.2.1 2D Zoom Star

Dans la représentation en deux dimensions, les axes sont reliés en fonction des valeurs que prennent les variables.

Prenons l'exemple de la figure 7.1. Le jeu de données est issu du site Sodas et est téléchargeable à l'adresse

`http://www.ceremade.dauphine.fr/~touati/exemples.htm`.

Il concerne la description des pays sous-développés du continent asiatique par 7 variables dont

- 4 variables de type intervalle (population, superficie, espérance de vie et analphabétisme) ;
- 3 variables modales (code langue 2, nature régime et code religion).

Sur cette représentation graphique, nous avons les informations suivantes :

- si la variable est quantitative ou catégorique classique, sa valeur est donnée ;
- si la variable est multivaluée, les différentes valeurs que prend la variable sont renseignées ;
- si la variable est de type intervalle, le minimum et le maximum de l'intervalle sont donnés ;
- si la variable est modale, les distributions de fréquence sont indiquées pour chaque catégorie.

Dans notre exemple, figure 7.1, nous pouvons lire, parmi les pays sous-développés du continent asiatique, que la religion principale est l'islamisme, la deuxième langue principale est l'anglais et que le taux d'analphabétisme se situe entre 4.80 et 54.60.

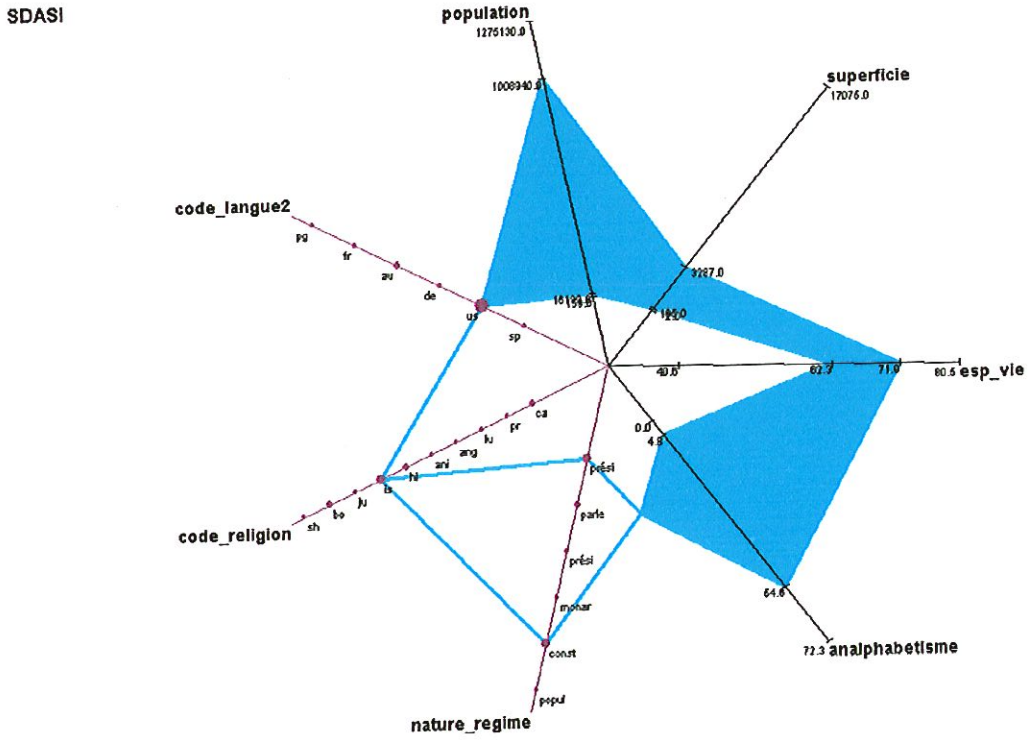


figure 7.1: Exemple d'une représentation Zoom Star en deux dimensions

Sur le graphique, sont reliés par une ligne :
- les valeurs que prennent les variables classiques,
- les différentes valeurs que prennent les variables multivaluées,
- les limites de l'intervalle pour des variables de type intervalle et la surface ainsi délimitée est remplie,
- la valeur ayant le plus grand poids (ou probabilité ou fréquence) pour les variables modales. Dans le cas où plusieurs catégories ont le même poids, elles sont toutes reliées.

Remarquons aussi qu'un axe correspondant à des valeurs manquantes sera grisé sur le graphique.

Le principal avantage de la représentation en deux dimensions est qu'elle fournit une image générale qui peut être associée à un objet symbolique. Remarquons aussi que cette méthode ne permet de visualiser qu'un seul objet à la fois.

Si nous voulons plus de précision sur la distribution des variables modales, il suffit de faire appel à la représentation en trois dimensions.

7.2.2 3D Zoom Star

Sur le Zoom Star en trois dimensions sont ajoutées les distributions correspondant à chaque variable modale.

Regardons ce que cela donne sur la figure 7.2. Cette figure représente le même objet que celui de la figure 7.1.

Sur la représentation en trois dimensions, les axes ne sont pas reliés.

L'avantage de la représentation en trois dimensions est de fournir des informations plus détaillées.

7.3 Construction des hiérarchies de partitions

Comme nous l'avons vu au chapitre 3, les critères agglomératifs des quatre méthodes de classification hiérarchiques se basent sur une matrice de distances ou de dissimilarités entre les individus.

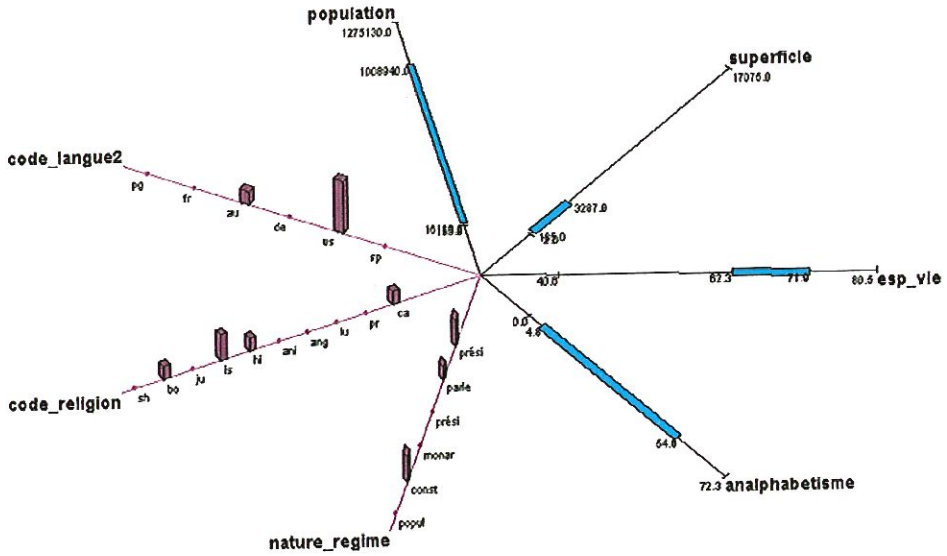


figure 7.2: Exemple d'une représentation Zoom Star en trois dimensions

Cette matrice de distances est calculée de la manière décrite à la fin du chapitre 2. Cependant, il faut noter une légère différence de façon à s'aligner sur le programme Sclust. Cette différence se situe au niveau du calcul de la distance totale. La distance entre deux individus x_k et x_l est définie par

$$\begin{aligned} d: \quad & E \times E \quad \rightarrow \mathbb{R}^+ \\ & (x_k, x_l) \quad \rightsquigarrow d(x_k, x_l) = \sum_{j=1}^{p} \delta_j(x_{kj}, x_{lj}) \end{aligned}$$

où δ_j est une des dissimilarités définies à la fin du chapitre 2.

Grâce à cette matrice de distances, les hiérarchies de partitions peuvent être construites en fonction du critère agglomératif de la méthode de classification classique considérée.

7.4 Méthodes de détermination du nombre de classes de Milligan et Cooper

Les méthodes de détermination du nombre de classes de Milligan et Cooper ont été initialement programmées par A.D. Gordon pour être appliquées à quatre méthodes hiérarchiques. Le code en Fortran 77 de ce programme a été "traduit" en C++ par S. Delogne qui a de plus adapté les méthodes de détermination du nombre de classes de Calinski et Harabasz, du C-index et Gamma pour qu'elles puissent être appliquées à la fois aux hiérarchies de partitions générées par les quatre méthodes de classification classiques et aux partitions successives produites par la méthode de classification non-hiérarchique Sclust.

Ce code a été inséré dans le programme Sclust pour calculer les indices des cinq meilleures méthodes de Milligan et Cooper (Calinski et Harabasz, Duda et Hart, C-index, Gamma et Beale).

7.5 Implémentation des méthodes et présentation des résultats dans le fichier listing

Les méthodes de détermination du nombre de classes de Milligan et Cooper ont été implémentées et adaptées au programme Sclust pour des objets décrits par des variables multivaluées et modales. Le listing se trouve en annexe.

Regardons maintenant comment sont présentés les résultats obtenus par les méthodes de détermination du nombre de classes dans le fichier listing. Pour illustrer cela, nous allons nous baser sur les résultats fournis pour le jeu de données artificielles introduit au début du chapitre 6.

Pour commencer, ce sont les résultats concernant Sclust qui sont affichés.

```
=========================================================
MILLIGAN et COOPER sur les partitions générées par Sclust
=========================================================

 Ngps        C-H         C-index       Gamma
  3        6.40588       0.14696      0.60514
  2        9.72340       0.12680      0.87597
  1          -             -            -
```

Dans la première colonne est indiqué le nombre de classes contenues dans la partition sur laquelle les méthodes de détermination du nombre de classes de Milligan et Cooper ont été appliquées.
Dans les trois colonnes suivantes sont données les valeurs des indices des méthodes de Calinski et Harabasz, du C-index et Gamma. Rappelons que les méthodes de Duda et Hart ainsi que de Beale ne sont pas applicables aux méthodes non-hiérarchiques.

Ensuite, ce sont les résultats obtenus par les cinq méthodes de détermination du nombre de classes pour chacune des méthodes de classification hiérarchiques qui sont affichés.

```
=========================================================================
MILLIGAN et COOPER sur les partitions générées par 4 méthodes hiérarchiques
=========================================================================

Méthode du lien simple
----------------------
Ngps    Height  Leaders      C-H        D-H       C-index     Gamma      Beale
 11      0.000  10  12         -          -        0.00000    1.00000    0.00000
 10      0.000   3  11         -          -        0.00000    1.00000    0.00000
  9      0.000   2   8         -          -        0.00000    1.00000    0.00000
  8      0.333   4   7       28.85714    1.25011   0.00000    1.00000    0.00000
  7      0.333   2   4        8.70370    1.19160   0.03333    0.86486    1.75000
  6      0.500   6   9        9.10000    1.25011   0.03077    0.88430    0.00000
  5      0.500   5  10        9.06500    1.53106   0.03167    0.87698    0.00000
  4      0.500   3   6        6.43831    1.29639   0.09294    0.74202    2.25000
  3      0.500   2   3        5.12284    0.35020   0.17697    0.52709    0.52941
Partition   1  2  2  2  5  2  2  2  2  5  2  5
  2      0.500   1   2        9.72340   -0.90652   0.12680    0.74901    0.07251
  1      0.667   1   5          -       0.78465      -          -        0.81028
```

Méthode du centroïde

Ngps	Height	Leaders	C-H	D-H	C-index	Gamma	Beale
11	0.000	10 12	-	-	0.00000	1.00000	0.00000
10	0.000	3 11	-	-	0.00000	1.00000	0.00000
9	0.000	2 8	-	-	0.00000	1.00000	0.00000
8	0.333	4 7	28.85714	1.25011	0.00000	1.00000	0.00000
7	0.500	6 9	16.33333	1.25011	0.00862	0.98639	0.00000
6	0.500	1 3	12.53333	1.53106	0.01258	0.96923	0.00000
5	0.583	5 10	11.43902	1.53106	0.01538	0.92197	0.00000
4	0.667	2 4	10.62366	1.19160	0.02857	0.91008	1.75000
3	0.903	1 2	9.50503	0.58479	0.04985	0.83093	0.68289
Partition		1 1 1 1 5 6 1 1 6 5 1 5					
2	1.452	1 6	9.72340	0.32636	0.12680	0.74901	0.52330
1	2.451	1 5	-	0.78465	-	-	0.81028

Méthode de Ward

Ngps	Height	Leaders	C-H	D-H	C-index	Gamma	Beale
11	0.000	10 12	-	-	0.00000	1.00000	0.00000
10	0.000	3 11	-	-	0.00000	1.00000	0.00000
9	0.000	2 8	-	-	0.00000	1.00000	0.00000
8	0.333	4 7	28.85714	1.25011	0.00000	1.00000	0.00000
7	0.500	6 9	16.33333	1.25011	0.00862	0.98639	0.00000
6	0.667	1 4	12.53333	0.78240	0.01887	0.91899	0.66667
5	0.778	5 10	11.43902	1.53106	0.02051	0.88571	0.00000
4	1.067	1 2	11.02104	0.57359	0.02974	0.91667	0.64000
3	2.171	1 3	9.50503	0.66574	0.04985	0.83093	0.75049
Partition		1 1 1 1 5 6 1 1 6 5 1 5					
2	3.188	1 6	9.72340	0.32636	0.12680	0.74901	0.52330
1	8.463	1 5	-	0.78465	-	-	0.81028

Méthode du lien complet

Ngps	Height	Leaders	C-H	D-H	C-index	Gamma	Beale
11	0.000	10 12	-	-	0.00000	1.00000	0.00000
10	0.000	3 11	-	-	0.00000	1.00000	0.00000
9	0.000	2 8	-	-	0.00000	1.00000	0.00000
8	0.333	4 7	28.85714	1.25011	0.00000	1.00000	0.00000
7	0.500	6 9	16.33333	1.25011	0.00862	0.98639	0.00000
6	0.500	1 3	12.53333	1.53106	0.01258	0.96923	0.00000
5	0.667	5 10	11.43902	1.53106	0.01538	0.92197	0.00000
4	1.000	2 4	10.62366	1.19160	0.02857	0.91008	1.75000
3	1.333	1 2	9.50503	0.58479	0.04985	0.83093	0.68289
Partition		1 1 1 1 5 6 1 1 6 5 1 5					
2	2.500	1 6	9.72340	0.32636	0.12680	0.74901	0.52330
1	4.000	1 5	-	0.78465	-	-	0.81028

La première colonne correspond toujours au nombre de classes de la hiérarchie de partitions sur laquelle les méthodes de détermination du nombre de classes de Milligan et Cooper ont été appliquées. La deuxième colonne `Height` indique la hauteur du palier auquel les individus `Leaders` de la troisième colonne ont été regroupés.
Les cinq dernières colonnes donnent les valeurs des indices des cinq méthodes de détermination du nombre de classes de Milligan et Cooper.

Notons aussi que la ligne `Partition` se place dans le tableau en fonction du nombre k de classes recherchées et indique pour chaque individu (les individus sont considérés en ordre croissant en fonction de leur numéro) un numéro correspondant à la classe dans laquelle il se situe.

Dans notre exemple, pour la méthode du lien complet, la ligne `Partition` correspond à une partition en trois classes. La première classe a le label 1, la deuxième le label 5 et la troisième classe est constituée des individus ayant le label 6. Nous pouvons donc lire que l'individu 1 appartient à la première classe ainsi que les trois suivants, que l'individu 5 appartient à la seconde classe ayant le label 5, et ainsi de suite.

Chapitre 8

Applications

8.1 Introduction

Maintenant que nous avons implémenté les méthodes de détermination du nombre de classes dans Sclust, nous allons les appliquer sur différents jeux de données.

Rappelons que nous avons appliqué les cinq meilleures méthodes de Milligan et Cooper sur les partitions générées par

- Sclust
- la méthode du lien simple
- la méthode du lien complet
- la méthode du centroïde
- la méthode de Ward.

Nous allons appliquer ces cinq méthodes sur quatre jeux de données.
Le premier concerne des boucles mérovingiennes datant du 6-8ème siècle après Jésus-Christ. Le second est un jeu de données artificielles reprenant les caractéristiques de 14 animaux. Ces deux jeux de données ne contiennent que des variables multivaluées.
Le jeu de données suivant décrit les ventes effectuées dans 13 magasins e-Fashion. Pour terminer, nous analyserons les résultats obtenus sur un jeu de données décrivant les habitudes de consommation dans 25 régions du Royaume-Uni. Ces deux derniers jeux contiennent uniquement des variables modales.

8.2 Présentation des résultats

Pour chacun des jeux de données, nous décrirons les résultats obtenus pour chacune des distances afin de pouvoir comparer ces dernières.

Nous présenterons tout d'abord les résultats fournis par les méthodes de détermination du nombre de classes de Milligan et Cooper sur Sclust. Ils seront donnés sous la forme d'un tableau à double entrée :

		M1	M3	M4
SCLUST				

TAB. 8.1 – Tableau utilisé pour présenter les résultats des méthodes de Milligan et Cooper

Ce tableau ne comprend qu'une seule ligne correspondant à la méthode Sclust. Les trois dernières colonnes correspondent aux méthodes de détermination du nombre de classes de Milligan et Cooper et plus précisément aux méthodes de Calinski et Harabasz (M1), du C-index (M3) et Gamma (M4) étant donné que les méthodes de Duda et Hart (M2) et de Beale (M5) ne sont pas applicables aux méthodes de classification non-hiérarchiques. Le chiffre contenu à l'intersection de cette ligne avec une des trois colonnes indique le nombre de classes suggéré par la méthode de détermination du nombre de classe considérée appliquée aux partitions générées par Sclust.

La deuxième colonne du tableau indique si Sclust retrouve les classes naturelles présentes dans le jeu de données. Un "+" exprime une classification correspondant aux classes naturelles et un "−" le cas contraire.

Ensuite, nous présenterons un tableau contenant les valeurs des cinq méthodes de détermination du nombre de classes de Milligan et Cooper pour différents nombres k de classes. Ce tableau sera de la même forme que le tableau 8.2.

Par exemple, la valeur du C-index lorsque les données sont partitionnées en cinq classes sera mentionnée à l'intersection de la ligne $k = 5$ et de la colonne M3.

Ensuite, pour les hiérarchies de partitions générées par les quatre méthodes de classification classiques, les tableaux seront de la même forme. Nous ajouterons cependant les colonnes correspondant aux indices de Duda et Hart (M2) et de Beale (M5). Nous obtenons alors les tableaux 8.3 et 8.4.

SCLUST	M1	M3	M4
$k = 8$			
$k = 7$			
$k = 6$			
$k = 5$			
$k = 4$			
$k = 3$			
$k = 2$			
$k = 1$			

TAB. 8.2 – Valeurs des méthodes de Milligan et Cooper pour Sclust

		M1	M2	M3	M4	M5
Saut minimum Saut maximum Centroïde Ward						

TAB. 8.3 – Tableau utilisé pour présenter les résultats des méthodes de Milligan et Cooper

Dans le tableau 8.3, chaque ligne correspond à une des quatre méthodes de classification hiérarchiques.

Pour terminer, nous choisirons l'une ou l'autre méthode de classification hiérarchique et pour chacune d'entre elles, nous détaillerons les valeurs des méthodes de détermination du nombre de classes de Milligan et Cooper grâce au tableau suivant.

Méthode de classification	M1	M2	M3	M4	M5
$k = 8$					
$k = 7$					
$k = 6$					
$k = 5$					
$k = 4$					
$k = 3$					
$k = 2$					
$k = 1$					

TAB. 8.4 – Valeurs des méthodes de Milligan et Cooper pour la méthode de classification considérée

Pour le premier jeu de données, le tableau utilisé sera plus complet que celui que nous venons de voir. Deux colonnes seront rajoutées. La première donnera la hauteur du palier à laquelle les deux individus mentionnés dans

la deuxième colonne sont regroupés. Ce tableau sera de la forme du tableau 8.5.

Méthode de classification	Height	Leaders	M1	M2	M3	M4	M5
$k = 8$							
$k = 7$							
$k = 6$							
$k = 5$							
$k = 4$							
$k = 3$							
$k = 2$							
$k = 1$							

TAB. 8.5 – Valeurs des méthodes de Milligan et Cooper pour la méthode de classification considérée

8.3 Animaux

Ce premier jeu de données est artificiel. Nous avons pris 14 animaux et nous les avons décrits de façon à distinguer à quelle espèce animale ils appartiennent. Le jeu de données de départ contient donc 14 animaux dont

- 5 mammifères (chat, cheval, chien, ours, vache),
- 4 poissons (requin, saumon, thon, truite) et
- 5 oiseaux (aigle, hirondelle, oie, pigeon, poule)

décrits par 9 variables multivaluées.

La première variable donne le mode de vie de ces animaux, c'est-à-dire s'ils sont sauvages, domestiques ou d'élevage. La deuxième fournit les modes de déplacement possibles et comprend quatre modalités : ailes, pattes, palmes et nageoires. La troisième variable décrit pour chaque animal sa race. Par exemple, la race du chien est le labrador. Cette variable contient 14 modalités. Il est évident que cette variable ne nous aidera pas pour notre problème de classification puisque chaque animal a sa propre race. Les quatrième et cinquième variables donnent respectivement le mode de reproduction (ovipare ou vivipare) et de respiration (poumons ou branchies). La sixième variable décrit le type de peau de chaque animal (poils, plumes ou écailles). La variable suivante dit dans quels milieux vit principalement l'animal (eau, terre, air). La huitième variable décrit le type d'alimentation (carnivore, granivore, herbivore, insectivore ou omnivore) et la dernière indique si ces animaux allaitent leurs petits, s'ils couvent leurs oeufs ou s'ils ne s'en occupent pas.

Le jeu de données complet se trouve en annexe.

Nous allons appliquer les méthodes de détermination du nombre de classes sur ce jeu de données en espérant retrouver les trois classes naturelles.

8.3.1 Distance de De Carvalho

Commençons par appliquer les méthodes de détermination du nombre de classes de Milligan et Cooper aux partitions générées par Sclust. Les résultats obtenus sont les suivants :

		M1	M3	M4
SCLUST	+	3	3	3

TAB. 8.6 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux partitions générées par Sclust

Sclust retrouve la bonne partition des données en trois classes et les trois méthodes de Milligan et Cooper ont l'air de bien fonctionner. En effet, les indices des différentes méthodes prennent les valeurs suivantes :

Sclust	M1	M3	M4
8	4.65561	0.02011	0.98259
7	2.95738	0.03763	0.98704
6	4.49806	0.06616	0.96549
5	4.69607	0.08213	0.93414
4	7.56759	0.02703	0.95179
3	**10.51819**	**0.00000**	**0.95869**
2	4.64353	0.20468	0.68847
1	-	-	-

TAB. 8.7 – Valeurs des indices de Milligan et Cooper pour Sclust

Appliquons maintenant les méthodes de détermination du nombre de classes de Milligan et Cooper aux hiérarchies de partitions générées par les quatre méthodes de classification classiques.

Notons tout d'abord que, par la suite, les animaux de notre jeu de données seront représentés par un numéro (voir tableau 8.8). La méthode du lien simple commence par regrouper les individus 10, 11, 12 et 13. Il s'agit des poissons. Ensuite, elle associe l'hirondelle avec le pigeon, le chien avec l'ours et le cheval avec la vache. Toujours à la même hauteur de la hiérarchie, elle regroupe le chat avec le chien et l'ours ainsi qu'avec le cheval et la vache. Après elle associe l'aigle avec l'hirondelle et le pigeon.

1	aigle
2	chat
3	cheval
4	chien
5	hirondelle
6	oie
7	ours
8	pigeon
9	poule
10	requin
11	saumon
12	thon
13	truite
14	vache

TAB. 8.8 – Numéros associés aux différents animaux

A ce dernier groupe, elle ajoute l'oie et la poule. Nous obtenons donc finalement les trois classes attendues.

Lien simple	Height	Leaders		M1	M2	M3	M4	M5
13	2.000	11	12	6.34821	2.92250	0.00000	1.00000	0.00000
12	2.000	10	11	6.83442	1.65343	0.00000	1.00000	0.75015
11	4.000	10	13	4.84524	2.15631	0.00000	1.00000	1.66713
10	4.000	5	8	4.83272	2.92250	0.00000	1.00000	0.00000
9	4.000	4	7	5.04989	2.92250	0.00000	1.00000	0.00000
8	4.000	3	14	5.44315	2.92250	0.00000	1.00000	0.00000
7	4.000	2	4	5.67194	1.92856	0.01250	0.98593	1.00021
6	4.000	2	3	5.70685	1.43330	0.02418	0.95146	1.00833
5	4.000	1	5	6.93793	1.65343	0.00606	0.97357	0.75015
4	5.500	1	6	8.34712	1.63612	**0.00000**	**1.00000**	1.04195
3	6.000	1	9	**10.51819**	1.56658	**0.00000**	**1.00000**	1.13090
2	10.500	1	2	7.15497	**3.00671**	0.03259	0.79510	**3.17976**
1	12.000	1	10	-	2.51950	-	-	2.26074

TAB. 8.9 – Valeurs des indices de Milligan et Cooper pour la méthode du lien simple

Les méthodes de détermination du nombre de classes retrouvent toutes la présence de ces trois classes dans les données. Notons cependant que les méthodes du C-index (M3) et Gamma (M4) atteignent leurs valeurs idéales pour trois et quatre classes. La classe supplémentaire quand nous passons de trois à quatre classes ne contient qu'un seul individu (la poule).
Le fait que ces deux méthodes atteignent respectivement les valeurs 0 et 1 indique que pour chacune des partitions, aucun objet n'est mal placé, dans le sens où un objet est mal classé si sa distance à la classe à laquelle il appartient est plus grande que sa distance à une autre classe. Ce qui peut se comprendre

intuitivement ici en remarquant que la poule est l'oiseau le plus différent des autres puisqu'il vit au sol et qu'il est d'élevage.

Les résultats obtenus pour les méthodes du lien complet, du centroïde et de Ward sont fort semblables à ceux obtenus pour la méthode du lien simple. Toutes les méthodes forment bien les classes recherchées. L'indice de Calinski et Harabasz (M1) atteint un maximum en trois classes. Les indices de Duda et Hart (M2) et de Beale (M5) n'atteignent pas les valeurs qui permettent de rejeter l'hypothèse de fusion des classes mais atteignent un maximum en $k = 2$. Les méthodes du C-index (M3) et Gamma (M4) atteignent pour toutes les méthodes hiérarchiques leurs valeurs idéales, c'est-à-dire respectivement 0 et 1, pour trois et quatre classes.

En résumé, nous obtenons les résultats suivants.

		M1	M2	M3	M4	M5
Lien simple	+	3	3	3,4	3,4	3
Lien complet	+	3	3	3,4	3,4	3
Centroïde	+	3	3	3,4	3,4	3
Ward	+	3	3	3,4	3,4	3

TAB. 8.10 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

8.3.2 Distance L_2

Les méthodes de détermination du nombre de classes de Milligan et Cooper appliquées aux partitions générées par Sclust donnent les résultats repris dans le tableau 8.11.

		M1	M3	M4
SCLUST	+	3	3,4	3

TAB. 8.11 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux partitions générées par Sclust

Pour commencer, nous constatons que Sclust retrouve bien les trois classes naturelles. Seule la méthode du C-index (M3) hésite entre trois et quatre classes. Les deux autres méthodes détectent trois classes dans les données.

SCLUST	M1	M3	M4
8	5.34694	0.00000	1.00000
7	3.45942	0.03763	0.96198
6	6.08000	0.02151	0.96607
5	6.79762	0.01210	0.96888
4	8.36798	**0.00000**	0.99872
3	**10.51533**	**0.00000**	**1.00000**
2	6.58696	0.09384	0.67246
1	-	-	-

TAB. 8.12 – Valeurs des indices de Milligan et Cooper pour Sclust

Remarquons que la classe supplémentaire quand nous passons de trois à quatre classes est formée d'un seul individu. Il s'agit de la poule qui est l'oiseau le plus différent des autres (l'aigle, l'hirondelle, le pigeon, l'oie) puisque, comme nous l'avons déjà mentionné, il vit au sol et est d'élevage.

Regardons maintenant les valeurs prises par les indices des cinq méthodes de détermination du nombre de classes de Milligan et Cooper appliquées à la hiérarchie de partitions générée par la méthode du centroïde. Les résultats sont repris dans le tableau suivant.

Centroïde	Height	Leaders		M1	M2	M3	M4	M5
13	2.000	11	12	6.25000	2.92250	0.00000	1.00000	0.00000
12	2.000	10	11	6.72727	1.65343	0.00000	1.00000	0.75015
11	4.000	10	13	4.76667	2.15631	0.00000	1.00000	1.66713
10	4.000	5	8	4.75214	2.92250	0.00000	1.00000	0.00000
9	4.000	4	7	4.96324	2.92250	0.00000	1.00000	0.00000
8	4.000	3	14	5.34694	2.92250	0.00000	1.00000	0.00000
7	4.000	1	5	5.92667	1.65343	0.00000	1.00000	0.75015
6	5.000	2	4	6.41758	1.92856	0.01075	0.99199	1.00021
5	5.000	1	6	7.29419	1.49740	0.00913	0.99325	0.91692
4	5.667	2	3	8.36798	1.43330	**0.00000**	**1.00000**	1.00833
3	7.250	1	9	**10.51533**	1.62399	**0.00000**	**1.00000**	1.18706
2	13.040	1	2	7.03967	**3.03494**	0.05045	0.79774	**3.23123**
1	15.300	1	10	-	2.48793	-	-	2.22431

TAB. 8.13 – Valeurs des indices de Milligan et Cooper pour la méthode du centroïde

Toutes les méthodes mettent en évidence une classification en trois classes. A nouveau, les méthodes du C-index (M3) et Gamma (M4) indiquent la présence de trois ou quatre classes.

Les trois autres méthodes hiérarchiques nous donnent des résultats semblables. Nous obtenons alors le tableau 8.14.

		M1	M2	M3	M4	M5
Lien simple	+	3	3	3,4	3,4	3
Lien complet	+	3	3	3,4	3,4	3
Centroïde	+	3	3	3,4	3,4	3
Ward	+	3	3	3,4	3,4	3

TAB. 8.14 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

En conclusion, les résultats obtenus pour la distance L_2 sont les mêmes que pour la distance de De Carvalho. A noter cependant que la méthode du C-index (M3) appliquée aux partitions générées par Sclust atteint sa valeur idéale pour trois et quatre classes.

8.3.3 Distance L_1

Regardons les résultats obtenus en appliquant les méthodes de détermination du nombre de classes aux partitions générées par Sclust.

La distance L_1 donne le même genre de résultats que les autres distances. Elle retrouve aussi la bonne partition en trois classes.

		M1	M3	M4
SCLUST	+	3	3,4,5	3,4,5

TAB. 8.15 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux partitions générées par Sclust

Les méthodes du C-index (M3) et Gamma (M4) indiquent la présence de trois, quatre ou cinq classes dans les données. Comme pour la distance L_2, la classe supplémentaire en passant de trois à quatre classes est formée de la poule. En ce qui concerne la partition en cinq classes, elle n'est pas du tout intéressante puisque la classe supplémentaire formée par Sclust ne contient aucun élément.

Regardons à présent les résultats obtenus par les méthodes de détermination du nombre de classes appliquées aux hiérarchies de partitions produites par les quatre méthodes de classification classiques.

Notons qu'en ce qui concerne la méthode du lien simple (voir tableau 8.17), les méthodes de détermination du nombre de classes indiquent toujours

SCLUST	M1	M3	M4
8	3.20594	0.05000	0.94231
7	5.42972	0.05000	0.94169
6	5.90857	0.05102	0.98415
5	7.06122	**0.00000**	**1.00000**
4	8.30570	**0.00000**	**1.00000**
3	**10.49177**	**0.00000**	**1.00000**
2	6.45967	0.09145	0.66218
1	-	-	-

TAB. 8.16 – Valeurs des indices de Milligan et Cooper pour Sclust

la présence des trois classes naturelles mais que les méthodes du C-index (M3) et Gamma (M4) atteignent respectivement 0 et 1 aussi en $k = 5$.

Lien simple	Height	Leaders		M1	M2	M3	M4	M5
13	2.000	11	12	6.43452	2.92250	0.00000	1.00000	0.00000
12	2.000	10	11	6.92857	1.65343	0.00000	1.00000	0.75015
11	4.000	10	13	4.91429	2.15631	0.00000	1.00000	1.66713
10	4.000	5	8	4.90354	2.92250	0.00000	1.00000	0.00000
9	4.000	4	7	5.12605	2.92250	0.00000	1.00000	0.00000
8	4.000	3	14	5.52770	2.92250	0.00000	1.00000	0.00000
7	4.000	2	4	5.76371	1.92856	0.01250	0.99294	1.00021
6	4.000	2	3	5.80490	1.43330	0.01770	0.98030	1.00833
5	4.000	1	5	7.06122	1.65343	**0.00000**	**1.00000**	0.75015
4	6.000	1	9	7.80830	2.04003	0.01449	0.99197	1.50041
3	6.000	1	6	**10.49177**	1.09861	**0.00000**	**1.00000**	0.74142
2	11.000	1	2	6.84682	**3.05755**	0.04776	0.79510	**3.27305**
1	12.000	1	10	-	2.43427	-	-	2.16337

TAB. 8.17 – Valeurs des indices de Milligan et Cooper pour la méthode du lien simple

La partition en cinq classes est décrite dans le tableau suivant.

numéro de la classe	animaux présents dans la classe
1	requin, saumon, thon, truite
2	aigle, hirondelle, pigeon
3	chat, cheval, chien, ours, vache
4	oie
5	poule

Cette partition peut aussi nous sembler correcte puisque les deux classes supplémentaires par rapport à la classification en trois classes contiennent chacune un animal. Il s'agit de la poule et de l'oie qui sont les deux oiseaux les plus différents des autres. En effet, l'oie a des palmes et vit au sol mais

peut faire de grands déplacements en volant et elle est herbivore. Quant à la poule, elle vit au sol et elle est d'élevage.

Lien complet	Height	Leaders		M1	M2	M3	M4	M5
13	2.000	11	12	6.43452	2.92250	0.00000	1.00000	0.00000
12	2.000	10	11	6.92857	1.65343	0.00000	1.00000	0.75015
11	4.000	10	13	4.91429	2.15631	0.00000	1.00000	1.66713
10	4.000	5	8	4.90354	2.92250	0.00000	1.00000	0.00000
9	4.000	4	7	5.12605	2.92250	0.00000	1.00000	0.00000
8	4.000	3	14	5.52770	2.92250	0.00000	1.00000	0.00000
7	4.000	1	5	6.13333	1.65343	0.00000	1.00000	0.75015
6	6.000	3	4	5.98442	1.90922	0.01942	0.98545	1.33370
5	6.000	2	3	7.06122	0.79579	**0.00000**	**1.00000**	0.54018
4	6.000	1	6	8.30570	1.76096	**0.00000**	**1.00000**	1.16699
3	8.000	1	9	**10.49177**	1.51299	**0.00000**	**1.00000**	1.08036
2	15.000	1	2	6.84682	**3.05755**	0.04776	0.79510	**3.27305**
1	18.000	1	10	-	2.43427	-	-	2.16337

TAB. 8.18 – Valeurs des indices de Milligan et Cooper pour la méthode du lien complet

Pour la méthode du lien complet, les méthodes du C-index (M3) et Gamma (M4) indiquent la présence de trois, quatre ou cinq classes. La partition en cinq classes est la même que pour le lien simple. Pour obtenir la partition en quatre classes, il suffit de regrouper l'oie avec la classe formée des autres oiseaux. Nous retrouvons donc à nouveau la partition où chaque espèce forme bien une classe et où la poule constitue une classe à elle seule. La partition en quatre classes est différente de celle obtenue pour le lien simple puisque pour cette dernière méthode, la quatrième classe n'était pas composée de la poule mais de l'oie.

Notons que la méthode du centroïde donne des résultats similaires à ceux obtenus par la méthode du lien complet et que la méthode de Ward fait de même avec la méthode du lien simple.

		M1	M2	M3	M4	M5
Lien simple	+	3	3	3,5	3,5	3
Lien complet	+	3	3	3,4,5	3,4,5	3
Centroïde	+	3	3	3,4,5	3,4,5	3
Ward	+	3	3	3,4	3,4	3

TAB. 8.19 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

Alors que les méthodes de Calinski et Harabasz (M1), de Duda et Hart (M2) et de Beale (M5) s'accordent pour dire qu'il y a trois classes dans les données, les méthodes du C-index (M3) et Gamma (M4) hésitent entre trois, quatre et cinq classes.

8.3.4 Conclusion

Signalons tout d'abord que toutes les méthodes de classification retrouvent la bonne partition des données en trois classes pour les trois distances.

Alors que les méthodes de Calinski et Harabasz (M1), de Duda et Hart (M2) et de Beale (M5) indiquent clairement la présence de trois classes, les méthodes du C-index (M3) et Gamma (M4) hésitent entre trois, quatre et cinq classes.

Notons que, bien que la classification en trois classes nous semble la plus normale, les classifications en quatre et cinq classes peuvent aussi avoir leur intérêt.

8.4 Boucles mérovingiennes datant du 6-8ème siècle après Jésus-Christ

Ce jeu de données est téléchargeable à l'adresse

`http://www-rocq.inria.fr/sodas/WP6/data/data.html`

et se trouve en annexe. Il contient 59 boucles mérovingiennes décrites par six variables multivaluées avec en tout 23 catégories (voir tableau 8.20).

Variables	Catégories
Fixation	clous de fer, bossettes de bronze, aucune
Bordures	bords mouvementés, répétition de motifs, frises géométriques
Damasquinure	bichrome, placage prédominant, incrustation dominante, monochrome argent
Fond	plaque d'argent, hachures, trame géométrique
Incrustation	filiforme, bande hachurée, bande pointillée, ruban plein
Plaque	arabesque, grande taille, dorsale carrée, motifs animaliers, tresse, ronde

TAB. 8.20 – Variables symboliques sur les boucles

Nous espérons retrouver dans ce jeu de données les deux classes principales (A et B) décrites par les archéologues. Une partition de ces données a été obtenue par Leredde en 1979 en utilisant une classification hiérarchique basée sur les connaissances des archéologues. Représentons chaque boucle par un numéro. Les deux classes sont les suivantes :

Classes	Individus présents dans la classe
A	3, 4, 5, 7, 9, 11, 14, 15, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59
B	1, 2, 6, 8, 10, 12, 13, 16, 18, 19, 29, 37, 38, 39, 40, 41, 42, 43, 44, 45

TAB. 8.21 – Description des deux classes obtenues par les archéologues

La première classe contient des individus dont la variable "damasquinure" prend les modalités "incrustation dominante" et "monochrome argent" et dont la variable "fond" prend principalement la modalité "hachures". Les individus de la seconde classe par contre prennent pour la première de ces deux variables les modalités "bichrome" et "placage prédominant" et pour la seconde variable "plaque d'argent".

Un exemple de boucles mérovingiennes de chaque classe est fournit à la figure 8.1.

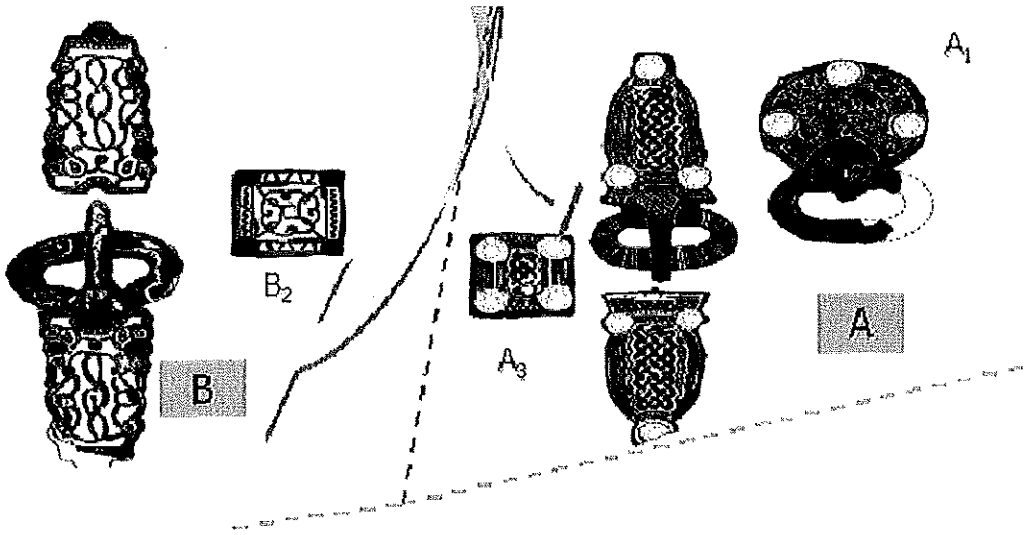


figure 8.1: Exemple de boucles mérovingiennes

Remarquons que les archéologues ont aussi défini une classification de ces individus en sept classes. Cette classification correspond en fait à une subdivision des deux classes ci-dessus. Elle est présentée dans le tableau suivant :

Classes	**Individus présents dans la classe**
A_1	15, 23, 47, 55, 56, 57, 58
A_2	5
A_3	3, 4, 9, 14, 20, 21, 22, 24, 25, 26, 28, 30, 31, 32, 33, 35, 36, 46, 49, 53
A_4	7, 11, 17, 27, 34, 48, 50, 51, 52, 59
B_1	8, 45
B_2	2, 6, 12, 13, 18, 19, 29, 44
B_3	1, 10, 16, 37, 38, 39, 40, 41, 42, 43

TAB. 8.22 – Description des sept classes obtenues par les archéologues

8.4.1 Distance de De Carvalho

Considérons la distance de De Carvalho et appliquons les méthodes de détermination du nombre de classes de Milligan et Cooper sur les partitions générées par Sclust. Les résultats obtenus sont repris dans le tableau 8.23.

		M1	M3	M4
SCLUST	–	2	2	2

TAB. 8.23 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

Sclust ne retrouve pas la partition en deux classes décrite par les archéologues. Cependant, il n'y a qu'un seul individu mal classé. Il s'agit de la boucle 20 qui semble avoir plusieurs caractéristiques communes à la fois avec les individus de la classe A et les individus de la classe B. En effet, comme la plupart des individus de la classe A, elle a comme type de fixation des clous de fer et la variable "damasquinure" prend les valeurs "incrustation dominante" et "monochrome argent" comme pour la plupart des individus de la classe B.

Remarquons que nous avons noté un "–" pour signaler que Sclust n'a pas retrouvé la partition des données fournie par les archéologues. Néanmoins, la partition fournie par Sclust n'est pas pour autant mauvaise.

Les indices des méthodes de Milligan et Cooper prennent les valeurs suivantes :

SCLUST	M1	M3	M4
8	28.52599	0.03049	0.95382
7	30.37194	0.04221	0.95325
6	23.36924	0.05127	0.92168
5	31.03050	0.04869	0.93004
4	32.66536	0.05707	0.95822
3	41.24019	0.03788	0.96124
2	**51.13007**	**0.01201**	**0.99917**
1	-	-	-

TAB. 8.24 – Valeurs des indices de Milligan et Cooper pour Sclust

La méthode de Calinski et Harabasz (M1) atteint un maximum absolu en deux classes. De même, les méthodes du C-index (M3) et Gamma (M4) présentent respectivement un minimum et un maximum pour ce même nombre de classes.

Appliquons maintenant les méthodes de détermination du nombre de classes de Milligan et Cooper sur les hiérarchies de partitions générées par quatre méthodes de classification classiques.

Les valeurs des indices des différentes méthodes de Milligan et Cooper sont présentées dans le tableau suivant en ce qui concerne la méthode du lien complet.

Lien complet	M1	M2	M3	M4	M5
8	31.07487	2.39966	0.00790	0.99238	3.72846
7	33.83576	2.35439	0.00860	0.99117	0.00000
6	34.66279	2.28551	0.01338	0.98565	2.34051
5	28.87829	3.56668	0.03270	0.94615	**3.59244**
4	34.53483	1.48947	0.03986	0.93230	1.20779
3	29.45371	2.91271	0.02761	0.95592	1.88482
2	**51.47556**	1.53622	**0.00932**	**0.99010**	1.24133
1	-	**5.27061**	-	-	**3.20213**

TAB. 8.25 – Valeurs des indices de Milligan et Cooper pour la méthode du lien complet

La méthode du lien complet retrouve bien la classification naturelle des données. La méthode de Calinski et Harabasz (M1) indique clairement la présence de deux classes dans les données. Les méthodes du C-index (M3) et Gamma (M4) atteignent aussi respectivement un minimum et maximum pour ce nombre de classes. L'indice de Duda et Hart (M2) dépasse la valeur de rejet en $k = 1$ et retrouve donc également deux classes. En ce qui concerne l'indice de Beale (M5), aucune des deux valeurs de rejet habituellement utilisées (4.61 ou 5.30) n'est atteinte. Cependant, il atteint deux valeurs fort proches et assez importantes en $k = 5$ et $k = 1$, ce qui nous laisse présumer la présence de deux ou six classes dans les données.

Il n'est pas très étonnant de retrouver une classification en six classes. En effet, nous avons vu que les archéologues avaient subdivisé les deux principales classes pour obtenir sept groupes. Quand nous observons les six classes fournies par la méthode du lien complet, l'une correspond exactement au groupe $A4$. Nous retrouvons presque aussi le groupe $A3$ à la seule différence que les éléments 20 et 35 sont absents. En effet, ils forment une classe à eux seuls. Les groupes $A1$ et $A2$ forment une seule classe ici. La méthode du lien complet divise la classe B en deux groupes mais aucun ne correspond à une des classes définies par les archéologues.

Les résultats obtenus sont résumés dans le tableau 8.26.

		M1	M2	M3	M4	M5
Lien simple	−	2	2	2	2	2
Lien complet	+	2	2	2	2	2 ou 6
Centroïde	−	2	2	2	2	2
Ward	+	2	2	2	2	2 ou 4

TAB. 8.26 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

Comme pour Sclust, les méthodes du lien simple et du centroïde ne retrouvent pas la classification naturelle des boucles puisqu'elles placent la boucle 20 dans le mauvais groupe.

8.4.2 Distance L_2

Commençons par appliquer les méthodes de détermination du nombre de classes de Milligan et Cooper aux partitions générées par Sclust.

Sclust ne retrouve pas la bonne partition en deux classes mais seul l'individu 20 se trouve dans un mauvais groupe. Les méthodes de Calinski et Harabasz (M1), du C-index (M3) et Gamma (M4) retrouvent toutes deux classes dans les données.

		M1	M3	M4
SCLUST	−	2	2	2

TAB. 8.27 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

En effet, les indices des différentes méthodes prennent les valeurs reprises dans le tableau 8.28.

Regardons maintenant ce que donnent les méthodes de détermination du nombre de classes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par les quatre méthodes de classification classiques.

Pour la méthode du lien complet, quelques différences sont à noter par rapport à la distance de De Carvalho.
Premièrement, la méthode ne retrouve plus la partition en deux classes décrite par les archéologues puisque l'individu 20 est placé dans la classe B au

SCLUST	M1	M3	M4
6	34.27613	0.05379	0.92349
5	24.55655	0.06042	0.93119
4	39.80523	0.03925	0.96119
3	41.26604	0.03353	0.94161
2	**46.39720**	**0.01376**	**0.98784**
1	-	-	-

TAB. 8.28 – Valeurs des indices de Milligan et Cooper pour Sclust

lieu de la classe A.
Deuxièmement, alors que les méthodes de Calinski et Harabasz (M1), de Duda et Hart (M2), du C-index (M3) et Gamma (M4) retrouvent toujours bien deux classes, la méthode de Beale (M5) aurait plutôt tendance à donner six classes.

Au niveau des six classes, l'une reprend les groupes $A1$ et $A2$, une autre correspond au groupe $A3$ sans l'individu 20 qui forme une classe à lui seul, la quatrième classe est le groupe $A4$ et les deux dernières divisent le groupe B mais nous ne pouvons distinguer aucun rapport avec la classification faite par les archéologues.

Lien complet	M1	M2	M3	M4	M5
8	29.96255	2.39966	0.00744	0.99387	3.72846
7	31.65414	0.81396	0.01461	0.98313	0.78073
6	34.63818	2.08800	0.01643	0.98027	2.24749
5	29.01056	4.40728	0.03381	0.94601	**6.59618**
4	31.86908	1.13986	0.05267	0.90653	0.97276
3	26.71250	2.99006	0.03080	0.94924	1.96205
2	**46.39720**	1.32172	**0.01376**	**0.98480**	1.08243
1	-	**4.90141**	-	-	2.88622

TAB. 8.29 – Valeurs des indices de Milligan et Cooper pour la méthode du lien complet

Nous obtenons les résultats repris dans le tableau 8.30.

Pour les méthodes du lien simple et du centroïde, toutes les méthodes s'accordent pour trouver deux classes dans les données malgré le fait que l'individu 20 ne soit pas placé dans la même classe que celle où les archéologues l'ont placé. Quant à la méthode de Ward, les résultats obtenus sont moins clairs, ce qui peut être dû au fait que non seulement l'individu 20 est mal classé mais également l'individu 35.

		M1	M2	M3	M4	M5
Lien simple	–	2	2	2	2	2
Lien complet	–	2	2	2	2	6
Centroïde	–	2	2	2	2	2
Ward	–	2	2	5	5	4

TAB. 8.30 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

8.4.3 Distance L_1

Voici les résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust.

		M1	M3	M4
SCLUST	–	2	2	2

TAB. 8.31 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

En effet, les indices des méthodes prennent les valeurs suivantes :

SCLUST	M1	M3	M4
6	34.27613	0.05379	0.92349
5	24.55655	0.06042	0.93119
4	39.80523	0.03925	0.96119
3	41.26604	0.03353	0.94161
2	**46.39720**	**0.01376**	**0.98784**
1	-	-	-

TAB. 8.32 – Valeurs des indices de Milligan et Cooper pour Sclust

Notons à nouveau que la partition retrouvée par Sclust n'est différente de la classification des archéologues qu'à un élément près. L'élément qui pose problème est toujours l'individu 20.

Regardons maintenant les résultats obtenus en appliquant les méthodes de Milligan et Cooper aux hiérarchies de partitions générées par les quatre méthodes de classification classiques.

Pour la méthode de Ward, l'indice de Beale (M5) met en évidence une classification en quatre classes. Cette partition est la suivante. La première classe reprend les groupes $A1$ et $A2$. La deuxième correspond au groupe $A3$.

La troisième classe est le groupe $A4$ et la dernière classe correspond au groupe B.

Ward	M1	M2	M3	M4	M5
8	28.02594	1.46563	0.02551	0.96836	1.32603
7	28.77218	1.69869	0.03059	0.95963	1.56108
6	30.63448	1.98742	0.03620	0.94777	2.05914
5	33.30967	1.39366	0.04886	0.92312	1.14068
4	36.56043	1.70319	0.04055	0.92956	1.36556
3	39.29677	3.44566	0.04663	0.92363	**3.38329**
2	**47.02353**	2.58917	**0.00771**	**0.99482**	1.66575
1	-	**4.94889**	-	-	2.92518

TAB. 8.33 – Valeurs des indices de Milligan et Cooper pour la méthode de Ward

Les résultats obtenus sont présentés dans le tableau suivant :

		M1	M2	M3	M4	M5
Lien simple	–	2	2	2	2	2
Lien complet	–	2	2	2	2	6
Centroïde	–	2	2	2	2	2
Ward	–	2	2	2	2	4

TAB. 8.34 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

Aucune des méthodes ne retrouve ici la classification donnée par les archéologues. A chaque fois, c'est l'individu 20 qui pose problème. Toutes les méthodes indiquent la présence de deux classes naturelles dans les données pour les méthodes du lien simple et du centroïde. Pour la méthode du lien complet, l'indice de Beale (M5) retrouve six classes.

Notons aussi que la méthode du centroïde place les éléments 20 et 35 dans le mauvais groupe en ce qui concerne la classification en deux classes.

8.4.4 Conclusion

La distance de De Carvalho est celle qui donne les meilleurs résultats pour cet exemple-ci car seuls deux cas ne déterminent pas le bon nombre de classes dans les données. Il s'agit de la méthode de Beale (M5) sur les partitions générées par les méthodes de Ward et du lien complet. Et c'est aussi la seule qui retrouve la partition en deux classes décrite par les archéologues.

La distance L_2 est celle qui fournit les moins bons résultats.

Notons aussi que la classification des boucles mérovingiennes par les archéologues n'est probablement pas basée sur les mêmes critères que ceux qu'utilisent Sclust et les quatre méthodes hiérarchiques. Ceci expliquerait pourquoi l'individu 20, bien que placé dans la classe A par les archéologues, se retrouve la plupart du temps dans la classe B.

Pour ce jeu de données sur les boucles mérovingiennes datant du 6-8ème siècle après Jésus-Christ, nous concluons à la présence de deux classes naturelles dans les données.

8.5 Magasins e-Fashion

Le jeu de données regroupe sur trois années (1999, 2000 et 2001) les ventes d'une chaîne de 13 magasins de vêtements et accessoires, répartis dans six pays différents. Ce jeu de données peut se trouver à l'adresse suivante :

`http://www.ceremade.dauphine.fr/~touati/exemples.htm`.

Les 13 individus sont les magasins de Paris 6ème, Lyon, Rome, Barcelone, Toulouse, Aix-Marseille, Madrid, Berlin, Milan, Bruxelles, Paris 15ème, Paris 8ème et Londres.

Les 8 différentes variables modales sélectionnées dans le jeu de données sont :

	Nom	Nombre de modalités
1	étiquette article	153
2	catégorie	31
3	famille du produit	12
4	étiquette couleur	160
5	gamme de couleur	17
6	mois	12
7	niveau de vente	5
8	numéro	5

Les variables "étiquette article", "famille du produit", "étiquette couleur", "gamme de couleur" et "catégorie" décrivent les articles vendus dans les magasins. La variable "étiquette article" peut prendre par exemple les modalités suivantes :

- Collier avec tiges longues,
- Echarpe en Viscose avec rectangles,
- Blouson Esprit Rivoli 5 poches,
- Veste Tobaggo Shetland,

et elle leur associe la proportion des ventes réalisées pour chaque article dans le magasin considéré.
La variable "famille produit" associe aussi la proportion des ventes mais en fonction des types de produits comme par exemple :

- Robes,
- Pulls,
- Sweats et T-shirts,
- Chemisiers...

Les variables "étiquette couleur", "gamme de couleur" et "catégories" indiquent aussi la proportion des ventes mais en fonction des couleurs des

articles vendus (Bleu Marine Oxford, Bleu Indigo, Gris Chine, ...), en fonction des gammes de couleurs (Bleu, Gris, Marron, Bordeaux, Rouge, ...) et en fonction des catégories des produits vendus (Bermudas ou Shorts, Pantalons Stricts, Habits - Soirée, Col Roulé, ...).
La variable "mois" indique la proportion des ventes effectuées durant chaque mois de l'année. La variable "numéro" peut prendre les valeurs 2, 3, 4, 5, 6 qui correspondent au numéro de la promotion qui a été effectuée (promotion dans le magasin, publicité à la radio, envoi de publicités par email, ...). Le niveau de vente reprend les possibilités suivantes : très faible, faible, nul, moyen et grand. Il indique donc les résultats de vente réalisés pour chaque magasin.

Notons aussi que nous ne connaissons rien sur une éventuelle classification naturelle des données.

8.5.1 Distance de De Carvalho

Les résultats obtenus par les méthodes de détermination du nombre de classes appliquées aux partitions produites par Sclust sont présentés dans le tableau 8.35.

	M1	M3	M4
SCLUST	2	4	1

TAB. 8.35 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

Alors que la méthode Gamma (M4) n'indique aucune structure dans les données, la méthode de Calinski et Harabasz (M1) met en évidence deux classes et celle du C-index (M3) quatre classes. En effet, les indices prennent les valeurs suivantes :

SCLUST	M1	M3	M4
6	1.79292	0.11571	0.25692
5	1.27726	0.24985	0.16712
4	2.79681	**0.09853**	0.34385
3	2.02706	0.33618	0.29278
2	**2.95234**	0.39883	0.35245
1	-	-	-

TAB. 8.36 – Valeurs des indices de Milligan et Cooper pour Sclust

Voici la partition en deux classes retrouvée par Sclust.

```
Classe :   1  Cardinal :      7
================================
 (0)  "e-Fashion Paris 6ème"   [0.3]  (5)  "e-Fashion Aix-Marseille" [0.4]
 (8)  "e-Fashion Milano"       [0.1]  (9)  "e-Fashion Bruxelles"     [0.3]
 (10) "e-Fashion Paris 15ème"  [0.3]  (11) "e-Fashion Paris 8ème"    [0.2]
 (12) "e-Fashion London"       [5.4]

Classe :   2  Cardinal :      6
================================
 (1) "e-Fashion Lyon"          [0.5]  (2) "e-Fashion Roma"          [2.8]
 (3) "e-Fashion Barcelona"     [0.6]  (4) "e-Fashion Toulouse"      [0.7]
 (6) "e-Fashion Madrid"        [0.7]  (7) "e-Fashion Berlin"        [0.7]
```

Rappelons que le chiffre entre crochets indique la distance entre l'individu et le prototype de la classe.

Remarquons que le magasin e-Fashion de Londres est fort éloigné du prototype de sa classe par rapport aux autres individus. Cette remarque est aussi valable pour l'e-Fashion de Rome.

Notons que si nous exécutons plusieurs fois le programme, les résultats obtenus sont à chaque fois différents puisque la partition générée par Sclust n'est pas la même.

Nous pouvons aussi obtenir les résultats suivants qui mettent en évidence deux et trois classes dans les données bien que les indices ne soient pas très prononcés.

SCLUST	M1	M3	M4
6	1.22695	0.02504	0.71321
5	1.43277	0.25379	0.55251
4	1.77120	0.18457	0.52646
3	2.78372	**0.16625**	**0.60160**
2	**2.97962**	0.39780	0.49897
1	-	-	-

TAB. 8.37 – Valeurs des indices de Milligan et Cooper pour Sclust

Appliquons maintenant les méthodes de détermination du nombre de classes de Milligan et Cooper sur les hiérarchies de partitions générées par les quatre méthodes de classification classiques.

Toutes les méthodes de Milligan et Cooper indiquent la présence de deux classes dans les données. Toutes les méthodes de classification retrouvent la même partition en deux classes. L'une de ces classes ne contient que le

magasin de Londres. Nous verrons plus tard pourquoi ce magasin se distingue de tous les autres.

Regardons les valeurs que prennent les différentes méthodes de Milligan et Cooper pour la méthode de Ward.

Ward	M1	M2	M3	M4	M5
12	13.37300	2.74619	0.00000	1.00000	0.00000
11	9.16062	2.74619	0.00000	1.00000	0.00000
10	8.49961	2.74619	0.00000	1.00000	0.00000
9	8.68826	2.74619	0.00000	1.00000	0.00000
8	8.68813	2.74619	0.00192	0.95616	0.00000
7	8.72421	2.62949	0.00406	0.91147	2.88679
6	8.83188	1.57477	0.00699	0.90769	1.20339
5	9.21637	1.86857	0.01319	0.86114	1.39562
4	10.09056	1.45145	0.01311	0.87364	1.20420
3	11.77580	1.40491	0.03753	0.74332	1.18708
2	**11.82885**	2.20615	**0.00000**	**1.00000**	2.00135
1	-	**3.33614**	-	-	**3.61685**

TAB. 8.38 – Valeurs des indices de Milligan et Cooper pour la méthode de Ward

Les résultats obtenus sont repris dans le tableau suivant :

	M1	M2	M3	M4	M5
Lien simple	2	2	2	2	2
Lien complet	2	2	2	2	2
Centroïde	2	2	2	2	2
Ward	2	2	2	2	2

TAB. 8.39 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

8.5.2 Distance L_2

Appliquons les différentes méthodes de Milligan et Cooper aux partitions générées par Sclust.

Les méthodes du C-index (M3) et Gamma (M4) retrouvent deux classes dans les données alors que l'indice de Calinski et Harabasz (M1) met en évidence quatre classes.

	M1	M3	M4
SCLUST	4	2	2

TAB. 8.40 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

Les indices des méthodes de Milligan et Cooper prennent les valeurs suivantes :

SCLUST	M1	M3	M4
6	12.82068	0.01958	0.92339
5	15.55857	0.00878	0.92339
4	**16.42842**	0.01586	0.86851
3	14.06922	0.00860	0.98370
2	9.37432	**0.00000**	**1.00000**
1	-	-	-

TAB. 8.41 – Valeurs des indices de Milligan et Cooper pour Sclust

Les deux classes générées par Sclust sont les suivantes :

```
Classe :   1  Cardinal :     12
==============================
"e-Fashion Paris 6ème"         "e-Fashion Lyon"
"e-Fashion Roma"               "e-Fashion Barcelona"
"e-Fashion Toulouse"           "e-Fashion Aix-Marseille"
"e-Fashion Madrid"             "e-Fashion Berlin"
"e-Fashion Milano"             "e-Fashion Bruxelles"
"e-Fashion Paris 15ème"        "e-Fashion Paris 8ème"

Classe :   2  Cardinal :      1
==============================
"e-Fashion London"
```

Regardons maintenant quelle est la partition en quatre classes.

```
Classe :   1  Cardinal :      4
==============================
"e-Fashion Toulouse"      "e-Fashion Berlin"
"e-Fashion Bruxelles"     "e-Fashion Paris 8ème"

Classe :   2  Cardinal :      1
==============================
"e-Fashion London"
```

```
Classe :   3  Cardinal :      6
==============================
"e-Fashion Paris 6ème"    "e-Fashion Lyon"
"e-Fashion Barcelona"     "e-Fashion Aix-Marseille"
"e-Fashion Milano"        "e-Fashion Paris 15ème"

Classe :   4  Cardinal :      2
==============================
"e-Fashion Roma"          "e-Fashion Madrid"
```

Nous verrons plus tard comment justifier cette classification. Nous allons voir en appliquant les méthodes de détermination du nombre de classes de Milligan et Cooper aux partitions générées par les quatre méthodes hiérarchiques que les magasins de Rome et de Madrid forment assez souvent une classe à eux deux.

Les méthodes de détermination du nombre de classes appliquées à la partition générée par la méthode du lien simple indiquent la présence de deux ou trois classes.

Lien simple	M1	M2	M3	M4	M5
12	13.66161	2.74619	0.00000	1.00000	0.00000
11	13.92453	2.74619	0.00000	1.00000	0.00000
10	10.92181	1.86632	0.00493	0.96759	1.39264
9	12.31303	2.74619	0.00333	0.97988	0.00000
8	14.03303	2.74619	0.00201	0.98929	0.00000
7	13.73140	2.17208	0.00403	0.95882	1.49983
6	6.74503	3.12887	0.05898	0.66558	4.16515
5	7.39445	0.71548	0.04560	0.78571	0.66439
4	9.29076	0.32172	0.01674	0.94209	0.43369
3	**14.06922**	2.74619	0.00860	0.98370	0.00000
2	9.37432	**3.17743**	**0.00000**	**1.00000**	**3.43713**
1	-	2.89458	-	-	2.86634

TAB. 8.42 – Valeurs des indices de Milligan et Cooper pour la méthode du lien simple

La partition en deux classes est toujours celle qui place le magasin de Londres dans un groupe et tous les autres magasins dans l'autre. La partition en trois classes est la suivante :

- e-Fashion Londres
- e-Fashion Rome et Madrid
- tous les autres magasins.

Pour les méthodes du lien complet, du centroïde et de Ward, nous trouvons la présence de deux, trois ou quatre classes dans les données.

Notons aussi que les quatre méthodes hiérarchiques retrouvent les mêmes partitions en deux et trois classes mais pas la même partition en quatre classes.

Nous obtenons donc les résultats suivants :

	M1	M2	M3	M4	M5
Lien simple	3	3	2	2	3
Lien complet	4	3	2	2	3
Centroïde	4	3	2	2	3
Ward	4	3	2	2	3

TAB. 8.43 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

8.5.3 Distance L_1

Appliquons maintenant les méthodes de Milligan et Cooper aux partitions générées par Sclust.

	M1	M3	M4
SCLUST	2	2	2

TAB. 8.44 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

Les trois méthodes retrouvent ici deux classes dans les données. Leurs indices prennent les valeurs reprises dans le tableau 8.45.

SCLUST	M1	M3	M4
6	2.73823	0.04145	0.68673
5	3.36401	0.03659	0.78747
4	3.92126	0.04104	0.85366
3	4.66519	0.04016	0.97147
2	**4.98088**	**0.00000**	**1.00000**
1	-	-	-

TAB. 8.45 – Valeurs des indices de Milligan et Cooper pour Sclust

Les deux classes sont les suivantes :

```
Classe :   1  Cardinal :     12
================================
"e-Fashion Paris 6ème"  "e-Fashion Lyon"          "e -Fashion Roma"
"e-Fashion Barcelona"   "e-Fashion Toulouse"     "e-Fashion Aix-Marseille"
"e-Fashion Madrid"      "e-Fashion Berlin"       "e-Fashion Milano"
"e-Fashion Bruxelles"   "e-Fashion Paris 15ème"  "e-Fashion Paris 8ème"

Classe :   2  Cardinal :      1
================================
"e-Fashion London"
```

Regardons à présent les résultats obtenus en appliquant les méthodes de détermination du nombre de classes de Milligan et Cooper aux hiérarchies de partitions générées par les quatre méthodes de classification classiques et, plus particulièrement, ceux obtenus pour la méthode du centroïde.

Centroïde	M1	M2	M3	M4	M5
12	2.25465	2.74619	0.00000	1.00000	0.00000
11	2.34730	2.74619	0.00000	1.00000	0.00000
10	2.47370	2.74619	0.00000	1.00000	0.00000
9	2.63188	2.74619	0.00000	1.00000	0.00000
8	2.83279	2.74619	0.00031	0.98356	0.00000
7	3.07308	2.74619	0.00282	0.95370	0.00000
6	3.27955	1.34993	0.01409	0.87941	0.84880
5	3.60827	0.86832	0.02052	0.86111	0.67643
4	3.93820	0.82997	0.02839	0.85278	0.72239
3	4.47216	0.78588	0.02744	0.90761	0.72785
2	**4.98088**	1.12212	**-0.00000**	**1.00000**	0.98654
1	-	**1.76553**	-	-	**1.52298**

TAB. 8.46 – Valeurs des indices de Milligan et Cooper pour la méthode du centroïde

Toutes les méthodes retrouvent deux classes dans les données et c'est à nouveau le magasin e-Fashion de Londres qui forme une classe à lui seul.

	M1	M2	M3	M4	M5
Lien simple	2	2	2	2	2
Lien complet	2	2	2	2	2
Centroïde	2	2	2	2	2
Ward	2	2	2	2	2

TAB. 8.47 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

8.5.4 Analyse

Pour la partition générée par Sclust, les distances L_1 et L_2 donnent deux classes alors que la distance de De Carvalho hésite entre deux, trois et quatre classes.

Pour les hiérarchies de partitions générées par les quatre méthodes de classification classiques, les distances de De Carvalho et L_1 donnent deux classes alors que les résultats fournis par la distance L_2 sont compris entre deux et quatre classes.

Remarquons que pour la classification en deux classes, une classe est composée d'un seul individu et l'autre de 12 individus. C'est le magasin se situant à Londres qui se distingue des autres. Pour mieux comprendre pourquoi, regardons les graphiques fournis par le logiciel Sodas (voir les figures 8.2 et 8.3). Remarquons que nous n'avons pas représenté ici toutes les variables sélectionnées. Les variables "étiquette article" et "étiquette couleur" ne sont pas représentables graphiquement puisqu'elles prennent beaucoup trop de modalités différentes et que le graphique devient alors illisible.

Par rapport au magasin e-Fashion Paris 6ème, nous observons que le magasin de Londres n'a utilisé qu'un type de promotion (variable "numéro"), n'a effectué des ventes qu'aux mois de novembre et décembre et n'a presque pas vendu d'accessoires mais plutôt des chemisiers, des sweats et des robes.

Deux autres magasins qui semblent se distinguer sont les magasins de Rome et Madrid. Nous pouvons voir sur les graphiques (voir figures 8.4 et 8.5) que ces deux magasins ne vendent pour ainsi dire que des accessoires, qu'ils ont fait principalement une promotion de type 4, que leurs ventes sont réparties de la même manière tout au long de l'année et qu'ils vendent surtout des articles de couleur bleue et noire.

8.5.5 Conclusion

D'après les résultats obtenus, nous pouvons conclure à la présence de deux classes dans les données, la première étant composée uniquement du magasin e-Fashion de Londres et la seconde comprenant tous les magasins restants.

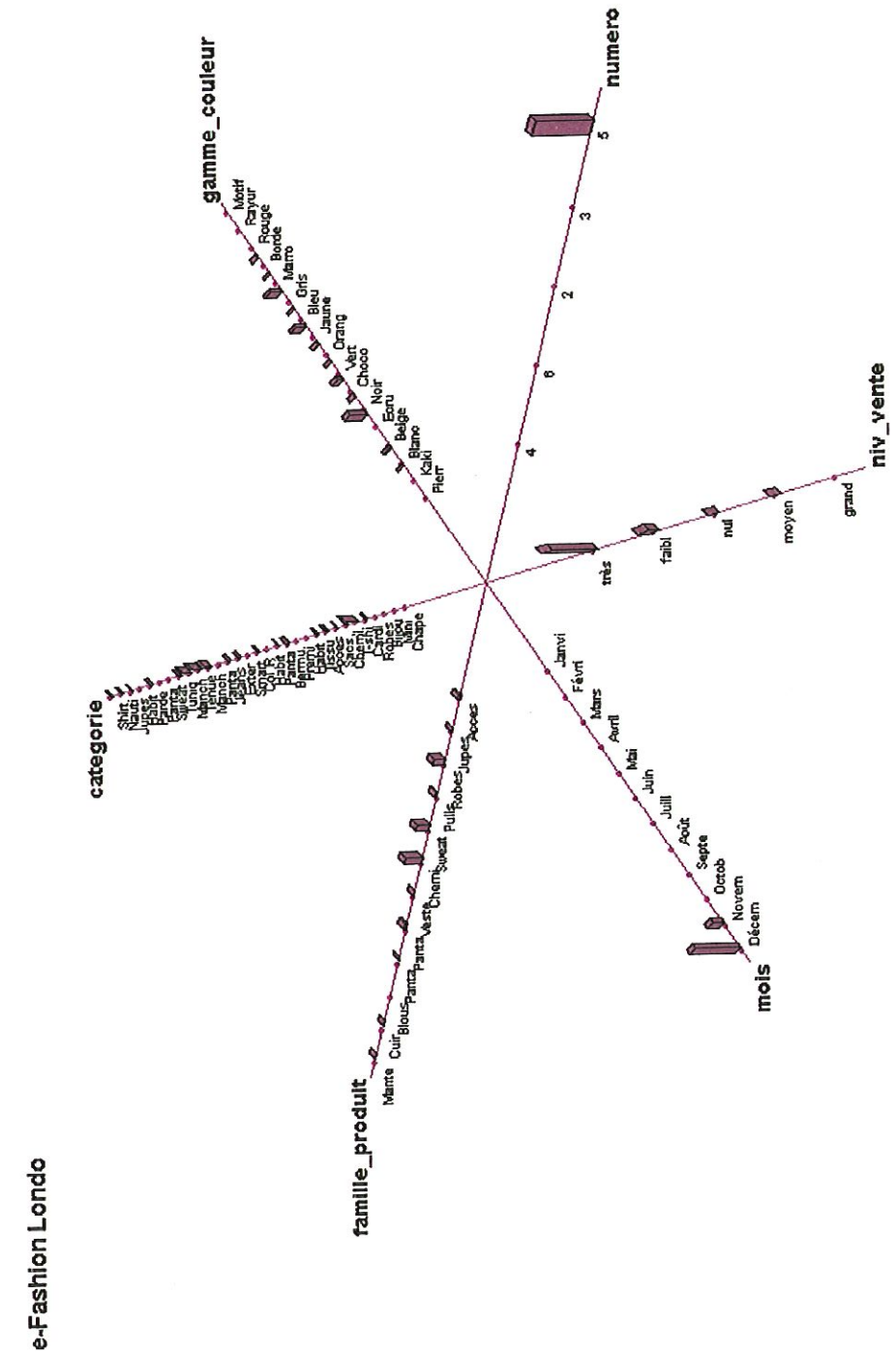


figure 8.2: e-Fashion Londres

figure 8.3: e-Fashion Paris 6ème

figure 8.4: e-Fashion Rome

figure 8.5: e-Fashion Madrid

8.6 Consommation

Ce jeu de données se trouve sur le cd de l'école Sodas de Porto ainsi qu'en annexe. Il décrit les habitudes de consommation des ménages au Royaume-Uni et contient 25 objets symboliques décrits par 14 variables modales. Les individus représentent 25 régions du Royaume-Uni :

- "Northern metropolitan"
- "North non-metropolitan"
- "Yorks and Humberside metropolitan"
- "Yorks and Humberside non-metropolitan"
- "East Midlands non-metropolitan"
- "North West metropolitan"
- "North West non-metropolitan"
- "South East other"
- "West Midlands metropolitan"
- "West Midlands non-metropolitan"
- "East Anglia"
- "Greater London North East"
- "Greater London North West"
- "Greater London South East"
- "Greater London South West"
- "South East metropolitan"
- "South West"
- "Wales i (Gwent, 3 Glamorgans)"
- "Wales ii (Clwyd, Gwynedd, Powys, Dyfed)"
- "Scotland i (Gram, High, Tay)"
- "Scotland ii (Loth, Fife, Cen)"
- "Scotland iii metropolitan (Strathclyde)"
- "Scotland iii non-metropolitan (Strath)"
- "Scotland iv (Dum/Gall, Bord)"
- "Northern Ireland"

La première variable indique si les habitants se chauffent grâce à un chauffage central. La seconde indique le type de chauffage central qu'ils utilisent (gaz de ville, combustible solide, électricité, mazout, gaz en bouteille, combustible solide et mazout, autre). La variable suivante dit s'ils possèdent une installation au chauffage central. La quatrième variable demande si le chauffage central a été réparé durant les 12 derniers mois. Les variables suivantes concernent

- les meubles (0, de 1 à 5, de 6 à 20, plus de 20)
- le mazout (0, de 1 à 10, plus de 10)

- les tapis (0, de 1 à 5, de 6 à 20, plus de 20)
- le téléphone (0, de 1 à 5, de 6 à 10, plus de 10)
- le gaz (0, de 1 à 5, de 6 à 10, de 11 à 20, plus de 20)
- l'électricité (0, de 1 à 5, de 6 à 10, de 11 à 20, plus de 20)
- les systèmes d'égouts (0, 1, 2, 3, plus de 3)
- la licence tv (0 ou 1, 2 ou 3)
- l'eau (0, 1 ou 2, 3, 4, 5, plus de 5).

La dernière variable concerne les appareils ménagers que possèdent les foyers (réfrigérateur séparé, machine à laver, lecteur de cd, micro-onde, congélateur séparé, sèche-linge, magnétoscope, lave-vaisselle, réfrigérateur-congélateur, aucun de ceux-ci).

Notons que nous ne connaissons rien sur la classification naturelle des données.

8.6.1 Distance de De Carvalho

Voici les résultats obtenus par les méthodes de détermination du nombre de classes de Milligan et Cooper appliquées aux partitions générées par Sclust.

	M1	M3	M4
SCLUST	3	5	4,5

TAB. 8.48 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

Les indices des méthodes de Milligan et Cooper prennent les valeurs suivantes.

SCLUST	M1	M3	M4
8	24.85787	0.03725	0.67755
7	13.01632	0.08131	0.57066
6	9.58830	0.11372	0.56170
5	8.89829	**0.08384**	**0.74487**
4	14.31491	0.10600	**0.74487**
3	**14.96634**	0.09364	0.70197
2	13.52756	0.17179	0.67913
1	-	-	-

TAB. 8.49 – Valeurs des indices de Milligan et Cooper pour Sclust

L'indice de Calinski et Harabasz (M1) atteint un maximum en trois classes. La méthode du C-index atteint un minimum en $k = 5$ et la méthode Gamma prend deux valeurs maximales en $k = 4$ et $k = 5$.

Regardons plus en détail les partitions en trois, quatre et cinq classes. La partition en trois classes est la suivante.

```
Classe :   1  Cardinal :      1
================================
( 24) "Northern ireland"                    [0.0]

Classe :   2  Cardinal :     17
================================
(  0) "Northern metropolitan"               [1.0]
(  1) "North non-metropolitan"              [0.7]
(  2) "Yorks and humberside metropoli"      [0.9]
(  3) "Yorks and humberside non-metro"      [0.1]
(  4) "East midlands non-metropolitan"      [0.0]
(  5) "North west metropolitan"             [1.0]
(  6) "North west non-metropolitan"         [0.5]
(  8) "West midlands metropolitan"          [2.7]
(  9) "West midlands non-metropolitan"      [0.1]
( 10) "East anglia"                         [0.0]
( 11) "Greater london north east"           [1.7]
( 12) "Greater london north west"           [2.2]
( 13) "Greater london south east"           [2.1]
( 14) "Greater london south west"           [2.1]
( 15) "South east metropolitan"             [0.0]
( 17) "Wales i (gwent, 3 glamorgans)"       [1.2]
( 18) "Wales ii (clw, gwy, pow, dyf)"       [0.5]

Classe :   3  Cardinal :      7
================================
(  7) "South east other"                    [0.0]
( 16) "South west"                          [0.0]
( 19) "Scotland i (gram, high, tay)"        [1.2]
( 20) "Scotland ii (loth, fife, cen)"       [1.5]
( 21) "Scotland iii met (strathclyde)"      [1.5]
( 22) "Scotland iii non-met (strath)"       [1.2]
( 23) "Scotland iv (dum/gall, bord)"        [1.6]
```

La partition en quatre classes est la suivante.

```
Classe :   1  Cardinal :     13
================================
(  0) "Northern metropolitan"              [0.7]
(  1) "North non-metropolitan"             [0.4]
(  2) "Yorks and humberside metropoli"     [0.5]
(  4) "East midlands non-metropolitan"     [0.0]
(  5) "North west metropolitan"            [0.6]
(  6) "North west non-metropolitan"        [0.3]
(  8) "West midlands metropolitan"         [2.4]
( 11) "Greater london north east"          [1.2]
( 12) "Greater london north west"          [2.0]
( 13) "Greater london south east"          [1.8]
( 14) "Greater london south west"          [2.1]
( 17) "Wales i (gwent, 3 glamorgans)"      [0.8]
( 18) "Wales ii (clw, gwy, pow, dyf)"      [0.3]

Classe :   2  Cardinal :      6
================================
( 19) "Scotland i (gram, high, tay)"       [0.1]
( 20) "Scotland ii (loth, fife, cen)"      [0.8]
( 21) "Scotland iii met (strathclyde)"     [0.8]
( 22) "Scotland iii non-met (strath)"      [0.0]
( 23) "Scotland iv (dum/gall, bord)"       [0.7]
( 24) "Northern ireland"                   [3.6]

Classe :   3  Cardinal :      3
================================
(  3) "Yorks and humberside non-metro"     [3.0]
(  7) "South east other"                   [0.0]
( 16) "South west"                         [0.0]

Classe :   4  Cardinal :      3
================================
(  9) "West midlands non-metropolitan"     [3.0]
( 10) "East anglia"                        [0.0]
( 15) "South east metropolitan"            [0.0]
```

Par contre, la partition en cinq classes fournie par Sclust n'est en fait qu'une partition en trois classes puisque deux des classes ne contiennent aucun élément. Cette partition correspond à la partition en quatre classes où les classes 1 et 4 ont été regroupées.

Les méthodes de détermination du nombre de classes de Milligan et Cooper donnent les résultats suivants pour la méthode du lien simple.

Lien simple	M1	M2	M3	M4	M5
8	14.54658	1.35041	0.05139	0.78007	1.11417
7	14.34058	1.48217	0.03567	0.87395	1.21390
6	17.92700	3.67962	0.03371	0.88414	0.00000
5	21.61340	**4.32994**	0.02770	0.91052	**7.98431**
4	26.72658	0.67673	0.00779	0.98402	0.67511
3	**36.11059**	3.54244	**0.00037**	**0.99944**	3.55533
2	10.58124	**8.19410**	0.00478	0.99124	**12.53151**
1	-	3.67471	-	-	3.02481

TAB. 8.50 – Valeurs des indices de Milligan et Cooper pour la méthode du lien simple

Les méthodes de Calinski et Harabasz (M1), du C-index (M3) et Gamma (M4) indiquent clairement la présence de trois classes dans les données. Par contre, celles de Duda et Hart (M2) et de Beale (M5) détectent une structure à deux niveaux différents de la hiérarchie. Bien qu'elles mettent en évidence une partition en six classes en atteignant leur valeur de rejet en $k = 5$, elles atteignent des valeurs fort importantes en $k = 2$ ce qui s'aligne sur les résultats obtenus par les autres méthodes.

La partition en trois classes est la suivante :
- Northern Ireland
- les cinq régions de Scotland
- toutes les autres régions.

La partition en six classes est donnée par :
- Greater London South West
- Northern Ireland
- Scotland iv (dum/gall, bord)
- Scotland i et iii non-metropolitan
- Scotland ii et iii metropolitan
- toutes les autres régions.

Pour la méthode de Ward, la méthode de Calinski et Harabasz (M1) n'indique aucune structure dans les données puisqu'elle prend des valeurs strictement décroissantes. Les méthodes du C-index (M3) et Gamma (M4) mettent en évidence trois classes. Les méthodes de Duda et Hart (M2) et de Beale (M5) dépassent leurs valeurs de rejet pour beaucoup de nombres de classes différents. En prenant un niveau de signification plus élevé, nous pouvons conclure que la méthode de Duda et Hart (M2) indique la présence de deux classes et la méthode de Beale (M5) la présence de trois classes.

Ward	M1	M2	M3	M4	M5
8	70.35186	5.09112	0.00565	0.94656	9.30310
7	61.73027	4.32994	0.00808	0.93150	7.98431
6	59.59217	3.27421	0.01172	0.91052	2.95709
5	53.21669	3.54244	0.01942	0.87314	3.55533
4	43.34432	5.04882	0.03812	0.82550	6.44227
3	36.11059	5.30172	**0.00037**	**0.99944**	5.72232
2	31.43970	5.17358	0.03810	0.95661	**12.01069**
1	-	**7.25129**	-	-	8.98753

TAB. 8.51 – Valeurs des indices de Milligan et Cooper pour la méthode de Ward

La partition en trois classes est la même que celle obtenue par la méthode du lien simple. La partition en deux classes est obtenue en regroupant Northern Ireland avec les régions de Scotland.

En résumé, nous avons obtenu les résultats suivants.

	M1	M2	M3	M4	M5
Lien simple	3	3,6	3	3	3,6
Lien complet	4	3,6	3	3	3,6
Centroïde	3	3	3	3	3,6
Ward	1	2	3	3	3

TAB. 8.52 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

Le nombre de classes qui revient le plus souvent est trois. Toutes les méthodes hiérarchiques retrouvent la même partition en trois classes. Nous regarderons plus tard si cette partition est logique ou non.

8.6.2 Distance L_2

Regardons maintenant les résultats obtenus par les méthodes de détermination du nombre de classes de Milligan et Cooper appliquées aux partitions générées par Sclust.

	M1	M3	M4
SCLUST	2	3,5	3

TAB. 8.53 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

En effet, les indices des méthodes de Milligan et Cooper prennent les valeurs reprises dans le tableau 8.54.

SCLUST	M1	M3	M4
8	13.89174	0.02812	0.73194
7	14.43556	0.02183	0.87461
6	11.21341	0.04871	0.84937
5	16.27142	**0.01075**	0.95751
4	15.04607	0.02088	0.94977
3	20.07273	**0.01142**	**0.97684**
2	**23.42443**	0.04876	0.92494
1	-	-	-

TAB. 8.54 – Valeurs des indices de Milligan et Cooper pour Sclust

Regardons la partition en deux classes.

```
Classe :   1  Cardinal :      7
================================
( 18) "Wales ii (clw, gwy, pow & dyf)"    [1.4]
( 19) "Scotland i (gram, high & tay)"     [0.1]
( 20) "Scotland ii (loth, fife & cen)"    [0.5]
( 21) "Scotland iii met (strathclyde)"    [0.9]
( 22) "Scotland iii non-met (strath)"     [0.7]
( 23) "Scotland iv (dum/gall & bord)"     [0.7]
( 24) "Northern ireland"                  [2.7]

Classe :   2  Cardinal :     18
================================
(  0) "Northern metropolitan"             [1.7]
(  1) "North non-metropolitan"            [0.8]
(  2) "Yorks and humberside metropoli"    [0.5]
(  3) "Yorks and humberside non-metro"    [0.8]
(  4) "East midlands non-metropolitan"    [0.2]
(  5) "North west metropolitan"           [0.5]
(  6) "North west non-metropolitan"       [0.7]
(  7) "South east other"                  [1.1]
(  8) "West midlands metropolitan"        [0.8]
(  9) "West midlands non-metropolitan"    [0.5]
( 10) "East anglia"                       [2.3]
( 11) "Greater london north east"         [1.1]
( 12) "Greater london north west"         [1.0]
```

```
( 13) "Greater london south east"          [1.5]
( 14) "Greater london south west"          [1.1]
( 15) "South east metropolitan"            [0.6]
( 16) "South west"                         [1.5]
( 17) "Wales i (gwent & 3 glamorgans)"     [1.4]
```

Regardons à présent la partition en trois classes. Cette partition n'est pas la même que celle obtenue pour la distance de De Carvalho. Ici, Northern Ireland ne forme plus une classe à lui seul et les différentes régions de Scotland ne sont plus toutes dans la même classe.

```
Classe :   1  Cardinal :      3
==============================
( 18) "Wales ii (clw, gwy, pow & dyf)"     [0.8]
( 23) "Scotland iv (dum/gall & bord)"      [0.9]
( 24) "Northern ireland"                   [1.3]

Classe :   2  Cardinal :     18
==============================
(  0) "Northern metropolitan"              [1.7]
(  1) "North non-metropolitan"             [0.8]
(  2) "Yorks and humberside metropoli"     [0.5]
(  3) "Yorks and humberside non-metro"     [0.8]
(  4) "East midlands non-metropolitan"     [0.2]
(  5) "North west metropolitan"            [0.5]
(  6) "North west non-metropolitan"        [0.7]
(  7) "South east other"                   [1.1]
(  8) "West midlands metropolitan"         [0.8]
(  9) "West midlands non-metropolitan"     [0.5]
( 10) "East anglia"                        [2.3]
( 11) "Greater london north east"          [1.1]
( 12) "Greater london north west"          [1.0]
( 13) "Greater london south east"          [1.5]
( 14) "Greater london south west"          [1.1]
( 15) "South east metropolitan"            [0.6]
( 16) "South west"                         [1.5]
( 17) "Wales i (gwent & 3 glamorgans)"     [1.4]

Classe :   3  Cardinal :      4
==============================
( 19) "Scotland i (gram, high & tay)"      [1.7]
( 20) "Scotland ii (loth, fife & cen)"     [0.3]
( 21) "Scotland iii met (strathclyde)"     [1.4]
( 22) "Scotland iii non-met (strath)"      [0.6]
```

Et pour terminer, regardons la partition en cinq classes.

```
Classe :   1  Cardinal :     17
================================
(  0) "Northern metropolitan"              [1.9]
(  1) "North non-metropolitan"             [0.9]
(  2) "Yorks and humberside metropoli"     [0.5]
(  3) "Yorks and humberside non-metro"     [0.9]
(  4) "East midlands non-metropolitan"     [0.3]
(  5) "North west metropolitan"            [0.5]
(  6) "North west non-metropolitan"        [0.7]
(  7) "South east other"                   [1.3]
(  8) "West midlands metropolitan"         [0.8]
(  9) "West midlands non-metropolitan"     [0.6]
( 11) "Greater london north east"          [1.1]
( 12) "Greater london north west"          [1.1]
( 13) "Greater london south east"          [1.6]
( 14) "Greater london south west"          [1.1]
( 15) "South east metropolitan"            [0.7]
( 16) "South west"                         [1.7]
( 17) "Wales i (gwent & 3 glamorgans)"     [1.5]

Classe :   2  Cardinal :      2
================================
( 10) "East anglia"                        [1.0]
( 18) "Wales ii (clw, gwy, pow & dyf)"     [1.0]

Classe :   3  Cardinal :      3
================================
( 20) "Scotland ii (loth, fife & cen)"     [0.6]
( 21) "Scotland iii met (strathclyde)"     [1.7]
( 22) "Scotland iii non-met (strath)"      [0.8]

Classe :   4  Cardinal :      1
================================
( 24) "Northern ireland"                   [0.0]

Classe :   5  Cardinal :      2
================================
( 19) "Scotland i (gram, high & tay)"      [1.0]
( 23) "Scotland iv (dum/gall & bord)"      [1.0]
```

Regardons les résultats obtenus par les méthodes de Milligan et Cooper sur les hiérarchies de partitions générées par les quatre méthodes de classification classiques.

Notons tout d'abord que les quatre méthodes de classification classiques donnent toutes des hiérarchies de partitions différentes. Nous n'allons détailler ici que les résultats obtenus avec la méthode du lien complet.

Lien complet	M1	M2	M3	M4	M5
8	14.27375	2.18835	0.01778	0.82715	1.74080
7	15.39926	3.67962	0.01890	0.81946	0.00000
6	16.62893	1.55936	0.02602	0.77966	1.22903
5	16.07007	2.34098	**0.00554**	**0.98156**	1.90418
4	**17.53602**	3.23728	0.00818	0.97868	2.25095
3	11.29891	**5.50747**	0.04169	0.90502	**5.83815**
2	16.90273	2.79801	0.04690	0.89275	1.92962
1	-	**5.15354**	-	-	4.83191

TAB. 8.55 – Valeurs des indices de Milligan et Cooper pour la méthode du lien complet

La partition en cinq classes retrouvée par les méthodes du C-index (M3) et Gamma (M4) pour la méthode du lien complet est la suivante :

- Northern Ireland
- Scotland iv (dum/gall, bord)
- Wales ii (clw, gwy, pow, dyf) et Scotland i
- Les trois autres régions de Scotland
- Toutes les autres régions.

La partition en deux classes retrouvée par la méthode de Beale est la suivante :

- Wales ii, Scotland i, Scotland iv et Northern Ireland
- Toutes les autres régions.

Les résultats obtenus par les méthodes de détermination du nombre de classes sont donnés dans le tableau suivant.

	M1	M2	M3	M4	M5
Lien simple	3	3	5	5	3
Lien complet	4	2,4	5	5	2,4
Centroïde	3	3	5	5	3
Ward	2	2	4	4	2

TAB. 8.56 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

Par rapport aux résultats obtenus avec la distance de De Carvalho, la partition en trois classes revient moins souvent. Les différentes méthodes de détermination du nombre de classes hésitent entre deux, trois, quatre et cinq classes. Remarquons aussi qu'elles n'indiquent pas de partition en trois classes pour les méthodes du lien complet et de Ward.

Notons de plus que, la plupart du temps, les méthodes hiérarchiques ne forment pas les mêmes hiérarchies de partitions avec la distance L_2 qu'avec la distance de De Carvalho.

8.6.3 Distance L_1

Les méthodes de détermination du nombre de classes appliquées aux partitions générées par Sclust indiquent deux, trois ou six classes présentes dans les données.

	M1	M3	M4
SCLUST	2	3	6

TAB. 8.57 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées à Sclust

En effet, les indices des méthodes de Milligan et Cooper prennent les valeurs suivantes.

SCLUST	M1	M3	M4
8	3.55806	0.05989	0.80307
7	3.55247	0.04319	0.93022
6	4.13831	0.05443	**0.95813**
5	4.04670	0.09781	0.93258
4	4.83627	0.08079	0.93634
3	5.26865	**0.03248**	0.90988
2	**7.32395**	0.06497	0.87922
1	-	-	-

TAB. 8.58 – Valeurs des indices de Milligan et Cooper pour Sclust

La partition en deux classes retrouvée par Sclust est la même que celle obtenue pour la distance L_2.

La partition en trois classes est la suivante.

```
Classe :   1  Cardinal :     18
==============================
(  0) "Northern metropolitan"              [1.4]
(  1) "North non-metropolitan"             [0.9]
(  2) "Yorks and humberside metropoli"     [0.7]
(  3) "Yorks and humberside non-metro"     [1.0]
(  4) "East midlands non-metropolitan"     [0.5]
(  5) "North west metropolitan"            [0.8]
(  6) "North west non-metropolitan"        [0.9]
(  7) "South east other"                   [0.9]
(  8) "West midlands metropolitan"         [0.9]
(  9) "West midlands non-metropolitan"     [0.7]
( 10) "East anglia"                        [1.4]
( 11) "Greater london north east"          [1.2]
( 12) "Greater london north west"          [1.1]
( 13) "Greater london south east"          [1.2]
( 14) "Greater london south west"          [1.2]
( 15) "South east metropolitan"            [0.8]
( 16) "South west"                         [1.1]
( 17) "Wales i (gwent & 3 glamorgans)"     [1.2]

Classe :   2  Cardinal :      1
==============================
( 24) "Northern ireland"                   [0.0]

Classe :   3  Cardinal :      6
==============================
( 18) "Wales ii (clw, gwy, pow & dyf)"     [1.4]
( 19) "Scotland i (gram, high & tay)"      [0.6]
( 20) "Scotland ii (loth, fife & cen)"     [0.8]
( 21) "Scotland iii met (strathclyde)"     [1.0]
( 22) "Scotland iii non-met (strath)"      [0.8]
( 23) "Scotland iv (dum/gall & bord)"      [1.3]
```

Comme pour la partition en trois classes pour la distance de De Carvalho, Northern Ireland forme une classe à lui seul et les cinq régions de Scotland sont regroupées dans la même classe mais au lieu d'être associées avec South East other et South West, elles sont associées à Wales ii.

Et pour terminer, regardons la partition en six classes.

```
Classe :   1  Cardinal :      4
==============================
( 19) "Scotland i (gram, high & tay)"      [1.2]
( 20) "Scotland ii (loth, fife & cen)"     [0.6]
( 21) "Scotland iii met (strathclyde)"     [1.3]
( 22) "Scotland iii non-met (strath)"      [0.9]
```

```
Classe :   2  Cardinal :      7
================================
(  3) "Yorks and humberside non-metro"    [0.9]
(  4) "East midlands non-metropolitan"    [0.8]
(  7) "South east other"                  [0.9]
(  9) "West midlands non-metropolitan"    [0.9]
( 14) "Greater london south west"         [1.4]
( 15) "South east metropolitan"           [1.0]
( 16) "South west"                        [1.0]

Classe :   3  Cardinal :      1
================================
( 10) "East anglia"                       [0.0]

Classe :   4  Cardinal :      1
================================
( 24) "Northern ireland"                  [0.0]

Classe :   5  Cardinal :      2
================================
( 18) "Wales ii (clw, gwy, pow & dyf)"    [1.0]
( 23) "Scotland iv (dum/gall & bord)"     [1.0]

Classe :   6  Cardinal :     10
================================
(  0) "Northern metropolitan"             [1.3]
(  1) "North non-metropolitan"            [1.0]
(  2) "Yorks and humberside metropoli"    [0.9]
(  5) "North west metropolitan"           [0.6]
(  6) "North west non-metropolitan"       [0.9]
(  8) "West midlands metropolitan"        [0.8]
( 11) "Greater london north east"         [1.0]
( 12) "Greater london north west"         [1.1]
( 13) "Greater london south east"         [1.1]
( 17) "Wales i (gwent & 3 glamorgans)"    [1.3]
```

Appliquons les méthodes de détermination du nombre de classes de Milligan et Cooper aux hiérarchies de partitions générées par les quatre méthodes de classification classiques.

Regardons les résultats obtenus pour la méthode de Ward. La méthode de Calinski et Harabasz (M1) retrouve deux classes dans les données comme l'indice de Beale (M5). La méthode de Duda et Hart (M2) hésite quant à elle entre deux et cinq classes et les méthodes du C-index (M3) et Gamma (M4) indiquent trois classes.

Ward	M1	M2	M3	M4	M5
8	3.73933	3.67962	0.05288	0.82469	0.00000
7	3.89299	1.72860	0.05563	0.80977	1.22778
6	4.16950	1.35534	0.05376	0.82272	1.02274
5	4.61519	1.04175	0.06500	0.79068	0.84437
4	4.75804	**2.86479**	0.08520	0.75442	1.55247
3	5.34879	1.29667	**0.03057**	**0.93773**	1.08079
2	**7.32395**	1.98761	0.06497	0.87639	1.45554
1	-	**2.67201**	-	-	**2.09366**

TAB. 8.59 – Valeurs des indices de Milligan et Cooper pour la méthode de Ward

La partition en cinq classes est la suivante :

1. Northern Ireland
2. Wales ii et Scotland iv
3. les quatre autres régions de Scotland
4. North non-metropolitan, Yorks and Humberside non-metropolitan, East Midlands non-metropolitan, South east other, West Midlands non-metropolitan, East anglia, South West
5. les autres régions.

Nous obtenons la partition en trois classes en regroupant les classes 1 et 2 et les classes 4 et 5.

Le premier groupe de la partition en deux classes reprend les classes 1, 2, 4 et 5 de la partition en cinq classes et le deuxième groupe correspond à la classe 3.

Les cinq meilleures méthodes de détermination du nombre de classes de Milligan et Cooper donnent les résultats suivants.

	M1	M2	M3	M4	M5
Lien simple	5	5	5	2	5
Lien complet	2	2,4	4	4	2
Centroïde	4	5	5	5	4
Ward	2	2,5	3	3	2

TAB. 8.60 – Résultats obtenus par les méthodes de Milligan et Cooper appliquées aux hiérarchies de partitions générées par quatre méthodes de classification classiques

Alors qu'en utilisant la distance de De Carvalho, la plupart des méthodes de détermination du nombre de classes indiquaient trois classes, avec la distance L_1, seules les méthodes du C-index (M3) et Gamma (M4) appliquées à la hiérarchie de partitions de Ward mettent en évidence trois classes.

8.6.4 Analyse

Les méthodes de détermination du nombre de classes indiquent la présence de une à six classes dans les données. Alors que pour la distance de De Carvalho, il semble y avoir trois classes (ou peut-être six), pour les distances L_2 et L_1, cela varie de deux à cinq.

Notons que les hiérarchies de partitions sont aussi fort changeantes selon la distance et les méthodes hiérarchiques utilisées. Cependant, en général, nous remarquons que Northern Ireland se distingue des autres régions ainsi que Scotland iv et Wales ii.

Regardons donc les Zoom Star en trois dimensions associés à ces individus.

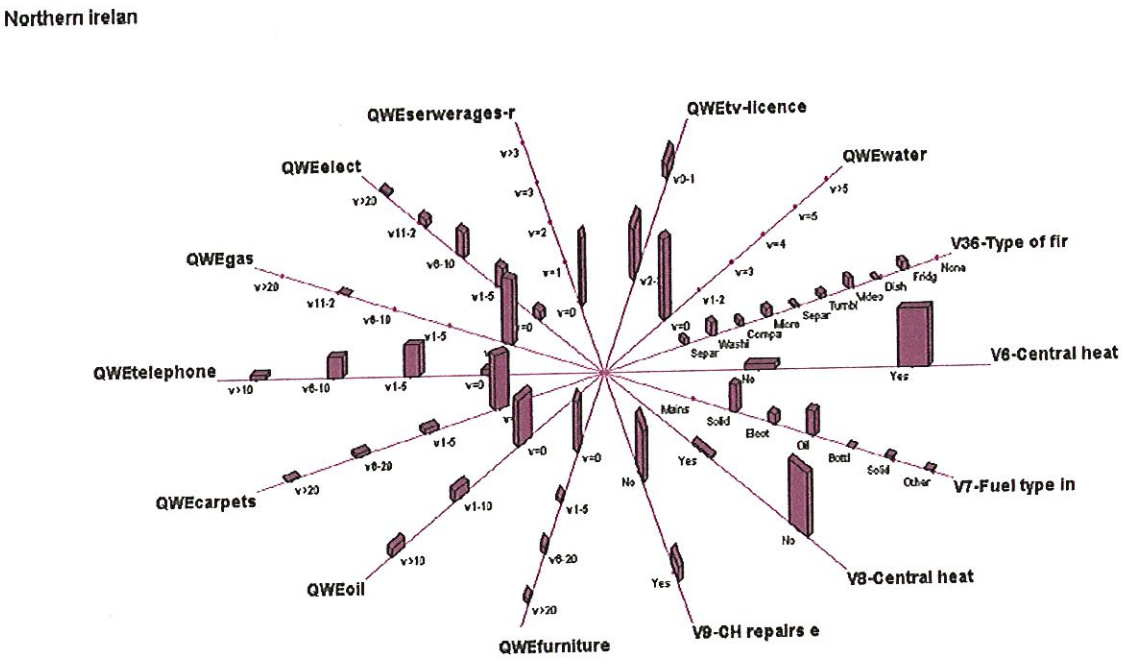


figure 8.6: Northern Ireland

Wales ii (clw,

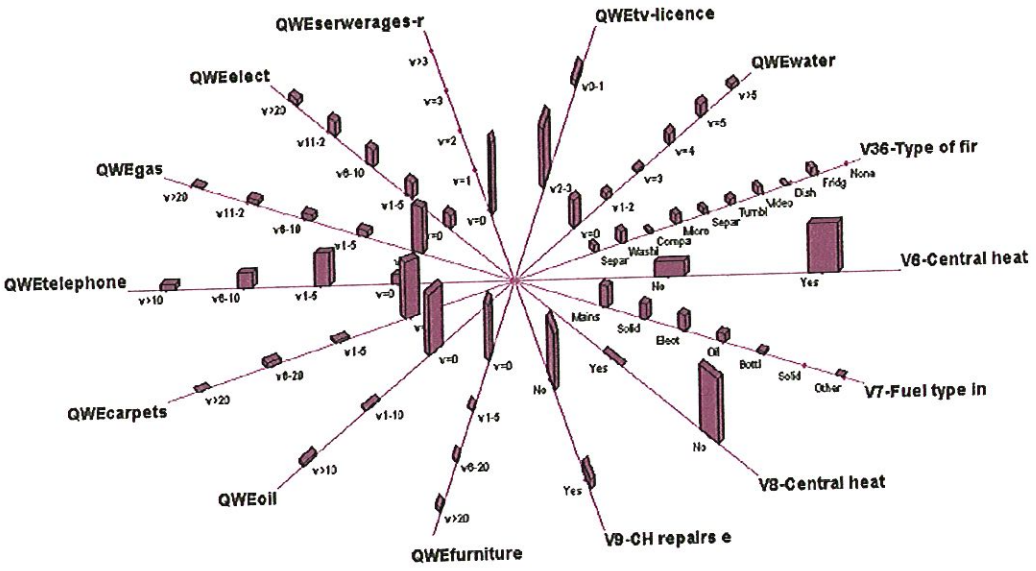


figure 8.7: Wales ii

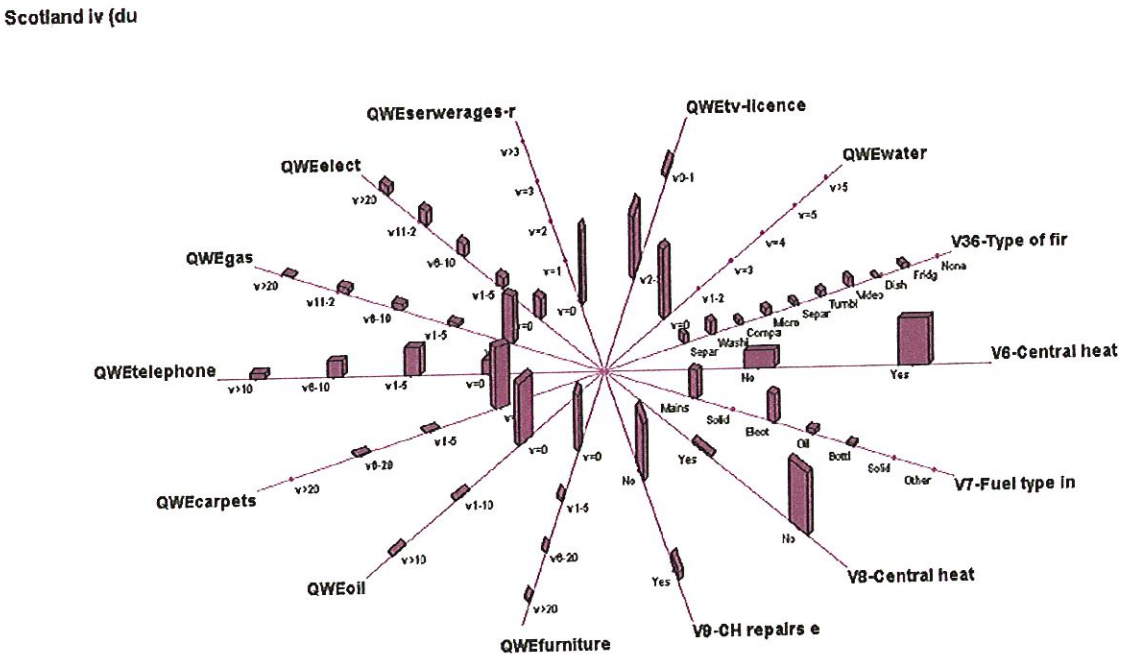


figure 8.8: Scotland iv

Greater london

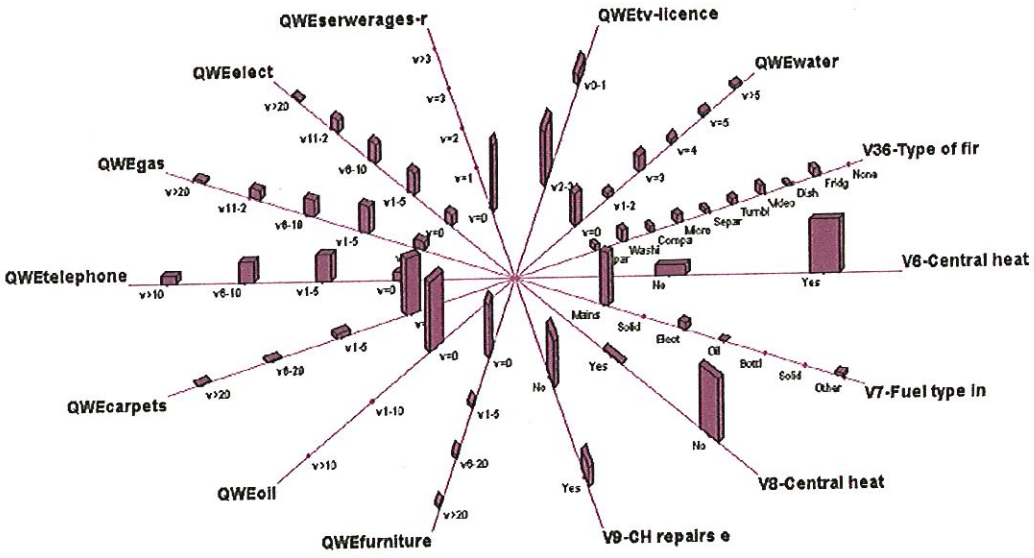


figure 8.9: Greater London South East

Il n'est pas très étonnant que Scotland iv et Wales ii se regroupent souvent puisqu'ils prennent des valeurs fort semblables pour la plupart des variables (sauf "eau" et "type de carburant").

Si nous comparons Northern Ireland avec Greater London South East, nous constatons que bien que certaines variables prennent des valeurs identiques, les variables "QWEoil", "Fuel type", "QWEwater" et "QWEgas" sont fort différentes. C'est aussi dû à ces quatre variables que Northern Ireland se distingue de Scotland iv et Wales ii.

Notons aussi que, d'une manière générale, les Zoom Star de tous les individus ont le même type d'allure.
Il est donc assez difficile de dire si une des partitions parmi celles retrouvées par Sclust et par les quatre méthodes hiérarchiques est meilleure qu'une autre.

8.6.5 Conclusion

Pour cet exemple, il est très difficile de se prononcer en faveur d'un nombre de classes plutôt qu'un autre. La grande variété des nombres de classes proposés par les différentes méthodes de détermination du nombre de classes vient du fait que les hiérarchies de partitions et partitions produites par les méthodes de classification sont souvent différentes.

8.7 Comparaison des distances

Pour chacun des exemples que nous venons de voir, nous avons présenté les résultats obtenus en fonction de la distance utilisée. Nous pouvons maintenant nous demander si une de ces trois distances (De Carvalho, L_1 ou L_2) donne des résultats plus intéressants.

En ce qui concerne les variables multivaluées, nous remarquons que la distance de De Carvalho est celle qui fournit les meilleurs résultats. Ce qui pourrait s'expliquer en remarquant que, contrairement aux distances L_1 et L_2, la distance de De Carvalho n'augmente que quand les individus ne prennent pas la même catégorie pour une certaine variable. Elle ne tient donc pas compte des différences des probabilités associées à une certaine catégorie lorsqu'une variable prend cette catégorie pour deux individus distincts.

En ce qui concerne les variables modales, il est plus difficile de dire, en se basant sur nos exemples, quelle distance semble la mieux adaptée à notre problème puisque nous ne connaissons rien sur une éventuelle classification naturelle des données.

8.8 Temps d'exécution

Reprenons sous forme d'un tableau les temps d'exécution obtenus pour les quatre jeux de données présentés précédemment. Les tests ont été effectués sur un AMD Athlon Thunderbird 700Mhz. Notons aussi que le temps calculé ne tient pas compte de l'initialisation de la structure `scluster`. Il indique donc juste le temps d'exécution de Sclust ainsi que des méthodes de détermination du nombre de classes sur les partitions générées par Sclust et sur les hiérarchies de partitions de quatre méthodes de classification classiques (lien simple, lien complet, centroïde et Ward). Notons aussi que nous n'avons pas changé les paramètres par défaut proposés par Sclust, c'est-à-dire que le nombre d'essais effectués par la méthode des nuées dynamiques est 5 et que le nombre d'itérations est de 20.

Le tableau 8.61 indique le temps en fonction du nombre de classes choisies. Le temps est exprimé en secondes.

La première chose qui apparaît en regardant ce tableau est que le troisième jeu de données prend un temps beaucoup plus important que celui sur les animaux alors qu'ils ont tous les deux un nombre comparable d'individus et de variables. Ceci peut sans doute être expliqué par le fait que

	3	4	5	6	7	8	9	10	11	12	13	14
boucles mérovingiennes	2	3	4	5	7	8	11	14	14	16	18	20
animaux	0	1	1	1	2	3	3	4	4	5	6	-
magasins e-Fashion	67	102	149	213	246	268	-	-	-	-	-	-
consommation	5	7	11	13	16	18	22	-	-	-	-	-

TAB. 8.61 – Temps d'exécution des différents jeux de données en fonction du nombre de classes recherchées (en secondes)

plusieurs variables prennent un nombre important de modalités (il s'agit des variables "étiquette article" avec 153 modalités et "étiquette couleur" avec 160 modalités).

En ce qui concerne le jeu de données sur les boucles mérovingiennes, le temps que nécessite le programme est un peu plus important que pour celui sur les animaux puisque le nombre d'individus considérés est plus important (59 contre 13).

Le jeu de données sur la consommation des ménages du Royaume-Uni prend un temps plus important que celui sur les boucles mérovingiennes bien que le nombre d'individus (25) soit inférieur. Ceci peut s'expliquer par le nombre de variables qui décrivent le jeu de données (14 contre 6).

D'une manière générale, nous pouvons dire que, pour des jeux de données contenant un nombre assez restreint d'individus et de variables (celles-ci contenant un nombre raisonnable de catégories ou modalités), les temps d'exécution sont assez performants mais que lorsqu'il augmente, le temps d'exécution devient assez vite important.

Conclusion

Le principal objectif de ce mémoire fut d'adapter les méthodes de détermination du nombre de classes au cas d'objets symboliques décrits par des variables multivaluées et modales. Nous avons appliqué ces méthodes aux partitions générées par la méthode de classification symbolique Sclust ainsi qu'aux hiérarchies de partitions générées par quatre méthodes de classification classiques.

Nous avons tout d'abord appliqué ces méthodes à deux jeux de données décrits par des variables multivaluées. Sur le premier jeu créé artificiellement, toutes les méthodes de classification classiques ont produit des hiérarchies de partitions contenant la structure naturelle des données. Les cinq meilleures méthodes de détermination du nombre de classes de Milligan et Cooper retrouvaient bien le nombre de classes naturelles présentes dans les données. De même, lorsque Sclust produisait des partitions comprenant la structure naturelle des données, les méthodes de Milligan et Cooper détectaient le bon nombre de classes. En ce qui concerne le second jeu de données sur les boucles mérovingiennes, les résultats furent relativement bons bien que plus diversifiés.

Ensuite, pour les deux jeux de données réelles décrits par des variables modales, les résultats furent moins concluants. Cela est dû au fait que les hiérarchies de partitions fournies par les diverses méthodes de classification étaient fort différentes. De plus, nous ne connaissions rien sur une éventuelle classification des données a priori.

Nous avons ensuite essayé de voir si une des trois distances utilisées semblait se distinguer des autres et permettait aux méthodes de classification de donner de meilleurs résultats. Alors que pour les variables multivaluées, la distance de De Carvalho semblait être plus intéressante, nous n'avons rien pu conclure pour les variables modales.

Pour terminer, nous avons analysé les temps d'exécution du programme Sclust et des méthodes de détermination du nombre de classes appliquées aux

partitions générées par sclust et aux hiérarchies de partitions générées par quatre méthodes de classification classiques. Il est ressorti de cette analyse que, pour de grands jeux de données ou pour des jeux de données décrits par des variables multivaluées ou modales prenant un grand nombre de modalités, l'exécution du programme prenait vite un temps assez important.

Aujourd'hui, les cinq meilleures méthodes de détermination du nombre de classes issues de l'étude de Milligan et Cooper ont donc été appliquées à des objets symboliques décrits par des variables intervalles, multivaluées ou modales. Une perspective serait de pouvoir les appliquer également lorsque les objets sont décrits par ces trois types de variables en même temps.

Annexe A

Listing du fichier classes2.cpp

```
#include "stdafx.h"
#include <fstream.h>
#include <iostream.h>
#include <math.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include "milligan.h"


/* ================================================================== */
void codage(FILE *lis,scluster para,CompSymbMat *M,int vsel[],
                                    double **x,double **d)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Codage de la matrice des données pour des variables multivaluées   */
/* et modales.                                                        */
/* Calcul de la matrice de distances.                                 */
/*                                                                    */
/* ================================================================== */
{
  int imax,kmax,nvar;
  int i,j,k,l,kk,mod_max,nmod,num,testmanq;
  double modalite,nb_mod;
  int i1;
  double tot,dd,ddt,total;

  imax=para.nb_objects;
  kmax=para.nb_class;
  nvar=para.nb_variables;
  mod_max=para.nb_mod;
```

```
// Initialisation
for (i=0;i<=imax;i++)
{
  for (j=0;j<=mod_max;j++)
  {
    x[i][j]=0.0;
  };
};

// Initialisation de la matrice de distances
for (i=1;i<=imax;i++)
{
  for (j=1;j<=imax;j++)
    d[i][j]=0.0;
};

for (i=1;i<=imax;i++) // Boucle sur les individus
{
  l=0;
  for (j=0;j<nvar;j++) // Boucle sur les variables
  {
    num=vsel[j];
    if (( M->index(i,num).na() || M->index(i,num).nu() )
          || (M->gettypevar(num) == continu
          ||  M->gettypevar(num) == intercont
          || M->gettypevar(num) == nominal))
          {
            testmanq=1;
            if(M->gettypevar(num) == continu || M->gettypevar(num) == nominal)
              l++;
            else
              if(M->gettypevar(num) == intercont)
                l=l+2;
              else
                l=l+M->getnbcateg(num);
          }
    else
          {
            testmanq=0;
            switch(M->gettypevar(num))
              {
                case continu:
                  {
                  };
                  break;
                case intercont:
                  {
                  };
                  break;
```

```
                case nominal: // Cas des variables nominales
                  {
                    nmod=M->getnbcateg(num);
                    for (kk=1;kk<=nmod;kk++)
                      {
                         l++;
                         if(kk==M->index(i,num).getcateg())
                           x[i][l]=1.;
                         else
                           x[i][l]=0.;
                      };
                  };
                  break;
                case multnomin: // Cas des variables multivaluées
                  {
                    nmod=M->getnbcateg(num);
                    nb_mod=0;
                    for(k=1;k<=nmod;k++)
                      {
                         modalite=M->index(i,num).getnomin()->index(k);
                         nb_mod=nb_mod+modalite;
                      };
                    for(k=1;k<=nmod;k++)
                      {
                         modalite=M->index(i,num).getnomin()->index(k);
                         l++;
                         x[i][l]=modalite/nb_mod;
                      };
                  };
                  break;
                case multnominm: // Cas des variables modales
                  {
                    nmod=M->getnbcateg(num);
                    for (k=1;k<=nmod;k++)
                      {
                         l++;
                         total=((dim_Mat*)M)->fuzzycard(i,num);
                         if( total == 0)
                           x[i][l]=M->index(i,num).getnominM()->indexm(k);
                         else
                           x[i][l]=M->index(i,num).getnominM()->indexm(k)/total;
                      };
                  };
                  break;
              };
          };
  };// Fin de la boucle sur les variables
}; // Fin de la boucle sur les individus

if(para.debug < 1)
```

```
{
  fprintf(lis," Matrice des données \n");
  for (i=1;i<=imax;i++)
    {
      fprintf(lis," %d ",i);
      for (j=1;j<=mod_max;j++)
        fprintf (lis," %f ",x[i][j]);
      fprintf(lis," \n ");
    };
};

// Calcul de la matrice de distances
for (i=1;i<=imax;i++) // Boucle sur les individus
  {
    i1=i-1;
    for (j=1;j<=i1;j++) // Boucle sur les individus
      {
        kk=0;
        tot=0.0;
        ddt=0.0;
        for(l=0;l<nvar;l++) // Boucle sur les variables
          {
            dd=0.0;
            num=vsel[l];
            nmod=M->getnbcateg(num);
            for (k=1;k<=nmod;k++)
              {
                kk++;
                switch(para.distance_modal) // Type de distance
                  {
                    case 0: // De Carvalho
                     if ((int)100.0*x[i][kk]==0.) dd=dd+x[j][kk];
                     if ((int)100.0*x[j][kk]==0.) dd=dd+x[i][kk];
                    break;
                    case 1: // L1
                     dd=dd+fabs(x[j][kk] - x[i][kk]);
                    break;
                    case 2: // L2
                     dd=dd+(x[j][kk] - x[i][kk])*(x[j][kk] - x[i][kk]);
                    break;
                    default :
                     if ((int)100.0*x[i][kk]==0.) dd=dd+x[j][kk];
                     if ((int)100.0*x[j][kk]==0.) dd=dd+x[i][kk];
                  };
              };
            ddt=ddt+dd;
          }; // Fin de la boucle sur les variables
        d[i][j]=ddt;
        d[j][i]=ddt;
      };// Fin de la boucles sur les individus j
```

```
    };  // Fin de la boucle sur les individus i

  if(para.debug < 1)
   {
     fprintf(lis,"\n MATRICE de dissimilarité \n");
     for (i=1;i<=imax;i++)
       {
         fprintf(lis," %d ",i);
         for (j=1;j<=imax;j++)
           fprintf (lis," %f ",d[i][j]);
         fprintf(lis," \n ");
       };
   };

};


/* ================================================================== */
void milligan_scluster_m(FILE *lis,scluster para,double **d,int **t_classe)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Calcul des indices des méthodes de détermination du nombre         */
/* de classes de Milligan et Cooper appliquées aux partitions         */
/* générées par SCLUST.                                               */
/*                                                                    */
/* ================================================================== */
{

  // Déclaration et initialisation des paramètres de la structure
  int imax,nvar,kmax;
  imax=para.nb_objects;
  nvar=para.nb_variables;
  kmax=para.nb_class;

  int i,j,k; // Variables locales, indices de boucles
  int ii,jj;
  int i1;
  double **BW; // Matrice liée au calcul de l'indice Gamma
    BW= new double* [imax+1];
    for (i=0;i<=imax;i++)
      BW[i]= new double [imax+1];
  double ttot=0.; // Inertie totale
  double *DLO=new double[imax*imax/2]; // Vecteur de distances
  int NC2;
  double ch=0.; // Indice de Calinski et Harabasz
  int *P=new int[imax+1]; // Vecteur tq P[i]=classe de l'individu i
  int numvar=1; // Numéro de la variable symbolique considérée
  int nbr=0; // Nombre de classes considérées
```

```
// Vecteur tq Nbr_obj[k]=nombre d'individus de la classe k
int *Nbr_obj=new int[kmax+1];
// Vecteur tq ssq[k]=inertie intra-classes de la classe k
double *ssq=new double[kmax+1];

// Variables liées au calcul de l'indice Gamma
int kin=0,kout=0;
double *BB=new double[imax*imax/2];
double *W=new double[imax*imax/2];
double u=0.0,dd=0.0;
int kk=0,ll=0;
double din=0.0;
double denmr=0.0;
double gamma=0.0; // Indice Gamma

// Variables liées au calcul du C-index
int *Nbr_comb=new int[kmax];
double c_min,c_max,c_index;
double total;
double intra;
double vec=0.;

fprintf(lis,"\n========================================================== \n");
fprintf(lis,"MILLIGAN et COOPER sur les partitions générées par Sclust \n");
fprintf(lis,"========================================================== \n \n");

// Initialisation de BW
for (i=2;i<=imax;i++)
  {
    i1=i-1;
    for (j=1;j<=i1;j++)
      BW[i][j]=0.0;
  }

ttot=0.0;
k=0;

// Calcul de ttot
for (i=1;i<=imax-1;i++)
  {
    for (j=i+1;j<=imax;j++)
      {
        k=k+1;
        DLO[k]=d[j][i];
        ttot=ttot+d[j][i];
      }
  };

ttot=ttot/double(imax); // ttot=Inertie TOTALE
```

```
NC2=imax*(imax-1)/2;

// Tri par ordre croissant des distances
// (de la plus petite à la plus grande)
for (i=1;i<=NC2-1;i++)
  {
    vec=DLO[i];
    for (j=i+1;j<=NC2;j++)
      {
        if (vec-DLO[j] > 0)
         {
           DLO[i]=DLO[j];
           DLO[j]=vec;
           vec=DLO[i];
         };
      };
  };

k=0;
ch=0.;

nbr=kmax;

fprintf(lis,"Ngps       C-H        C-index      Gamma\n \n");

for (nbr=kmax;nbr>=1;nbr--) // Boucle sur le nombre de classes
  {
    ssq[0]=0; // Initialisation inertie intra-classes
    ch=0.; // Initialisation indice C-H

    // Initialisation
    for (i=1;i<=imax;i++) P[i]=0;
    for (k=1;k<=kmax;k++) Nbr_obj[k]=0;
    for (k=1;k<=kmax;k++) Nbr_comb[k]=0;

    for (i=1;i<=imax;i++)
      {
        // P[i] contient la classe d'affectation de l'individu i
        P[i]=t_classe[i][nbr];
        // Nbr_obj[P[i]] contient le nombre d'objets
        // présents dans la classe P[i].
        Nbr_obj[P[i]]+=1;
      };

    for (k=1;k<=nbr;k++) ssq[k]=0.;

    // Calcul de l'inertie intra-classes
    for (k=1;k<=nbr;k++)
      {
        for (i=1;i<=imax-1;i++)
```

```
      {
        if (P[i]==k)
         {
           for (j=i+1;j<=imax;j++)
             {
               if (P[j]==k)
                {
                  ssq[k]+=d[i][j];
                };
             };
         };
      };
  };

intra=0.;
for (k=1;k<=nbr;k++)
intra+=ssq[k];

// Normalisation
for (k=1;k<=nbr;k++)
  {
    ssq[k]=ssq[k]/double(Nbr_obj[k]);
    if (Nbr_obj[k]==0)
      {
         ssq[k]=0;
      };
  };

// ssq[0]=somme des inerties intra-classes
ssq[0]=0;
for (k=1;k<=nbr;k++)
  ssq[0]+=ssq[k];

// Mise à jour de la matrice BW
for (k=1;k<=nbr;k++)
  {
    for (j=1;j<=imax;j++)
      {
        if (P[j]==k)
         {
           for (i=1;i<=imax;i++)
             {
               if (P[i]==k)
                {
                  ii=maxint(i,j);
                  jj=minint(i,j);
                  BW[ii][jj]=1.0;
                };
             };
         };
```

```
      };
  };

// Calcul de l'indice de C-H
if (nbr==1) ch=0.;
ch=(((ttot-ssq[0])*double(imax-nbr))/(ssq[0]*double(nbr-1)));

// Calcul de l'indice Gamma
kin=0;
kout=0;

for (i=2;i<=imax;i++)
  {
    ii=i-1;
    for (j=1;j<=ii;j++)
      {
        if (BW[i][j]<=0.0)
         {
           kout=kout+1;
           BB[kout]=d[i][j];
         }
        else
         {
           kin=kin+1;
           W[kin]=d[i][j];
         };
      };
  };

u=0.;
dd=0.;

for (kk=1;kk<=kin;kk++)
  {
    din=W[kk];
    for (ll=1;ll<=kout;ll++)
      {
        if ((din-BB[ll]) ==0) dd=dd+0.5;
        if ((din-BB[ll]) > 0) u=u+1.;
      };
  };

denmr=(double(kin)*double(kout)/2.)-dd;

gamma=(1.-u/denmr);

// Calcul du C-index
for (i=1;i<=nbr;i++) Nbr_comb[i]=((Nbr_obj[i])*(Nbr_obj[i]-1)/2);

Nbr_comb[0]=0;
```

```
      for (k=1;k<=nbr;k++) Nbr_comb[0]+=Nbr_comb[k];
      total=Nbr_comb[0];

      c_min=0.;
      c_max=0.;
      c_index=0.;

      for (j=1;j<=total;j++)
        {
          c_min=c_min+DLO[j];
          c_max=c_max+DLO[NC2-Nbr_comb[0]+j];
        };

      c_index=((intra-c_min)/(c_max-c_min));

      // Edition des résultats
      if (nbr!=1) fprintf(lis," %2d    %10.5f     %8.5f     %8.5f \n",
                          nbr,ch,c_index,gamma);
      else fprintf(lis," %2d          -              -            -\n\n",nbr);

    }; // Fin boucle sur le nombre de classes

  delete DLO;
  delete P;
  delete Nbr_obj;
  delete ssq;

};

/* ================================================================== */
void milligan_m(FILE *lis,scluster para,double **d)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Calcul des indices des méthodes de détermination du nombre         */
/* de classes de Milligan et Cooper appliquées aux hiérarchies        */
/* de partitions générées par 4 méthodes de classification classiques:*/
/*      - Méthode du lien simple                                      */
/*      - Méthode du centroïde                                        */
/*      - Méthode de Ward                                             */
/*      - Méthode du lien complet                                     */
/*                                                                    */
/* ================================================================== */
{
  // Initialisation des paramètres de la structure
  int imax,nvar,mod_max,kmax;
  int i,j,l;
  imax=para.nb_objects;
  nvar=para.nb_variables;
```

```
mod_max=para.nb_mod;
kmax=para.nb_class;

const double pi = 3.14159265 ;
const double eps = 1.e-6;

double ssq=0.0,dw=0.0;
int nclust,i1;

// Matrice de distances intermédiaire
double **DT;
   DT= new double* [imax+1];
   for (i=0;i<=imax;i++)
     DT[i]= new double [imax+1];

// Matrice liée au calcul de l'indice Gamma
double **BW;
   BW= new double* [imax+1];
   for (i = 0;i<=imax;i++)
     BW[i]= new double [imax+1];

double ttot=0.0; // Inertie totale
double *DLO=new double[imax*imax/2]; // Vecteur de distances
double *DHI=new double[imax*imax/2]; // Vecteur de distances
int NC2;
int N1;
double ch=0.0; // Indice de Calinski et Harabasz
double wtot=0.0; // Inertie intra-classes
double dtot=0.0;
int ndtot=0;
double btot=0.0; // Inertie inter-classes
double cind=0.0; // Indice du C-index

// Vecteur tq P[i]=classe de l'individu i
int *P=new int[imax+1];
// Vecteur tq Q[i]= nombre d'individus dans la classe i
int *Q=new int[imax+1];
double dmax=0.0;
double t=0.0;
int ic=0,jc=0;
double dw1=0.0,dw2=0.0;
double ssq1=0.0,ssq2=0.0;
double dwt=0.0,ssqt=0.0;
int nw1=0,nw2=0,nwt=0;
int ii=0,jj=0;
int *A=new int[imax+1];
int *B=new int[imax+1];
double *H=new double[imax+1]; // Niveaux de la hiérarchie
int qic=0,qjc=0,z=0;
double ff=0.0;
```

```
int qi=0;
int k1=0,k2=0;
double dj=0.0,dk=0.0;
int nqc=0,nw=0;
double fq=0.0,fp=0.0;
int ngp=0;
int numvar=0;

// Variables liées au calcul de l'indice de Beale
double beale=0.0,p1=0.0;
double denom=0.0;

// Variables liées au calcul de l'indice de Duda et Hart
double hold=0.0;
double hold2=0.0;

// Variables liées au calcul de l'indice Gamma
int kin=0,kout=0;
double *BB=new double[imax*imax/2];
double *W=new double[imax*imax/2];
double u=0.0,dd=0.0;
int kk=0;
double din=0.0;
int ll=0;
double denmr=0.0;
double gamma=0.0; // Indice Gamma


fprintf(lis,"\n=========================================================
             ========== \n");
fprintf(lis,"MILLIGAN et COOPER sur les partitions générées par 4
             méthodes hiérarchiques \n");
fprintf(lis,"=========================================================
             ========== \n \n");


for(nclust=1;nclust<=4;nclust++) // Choix d'une méthode de classification
  {
    for (i=0;i<=imax;i++)
      for (j=0;j<=imax;j++)
        {
          DT[i][j]=d[i][j];
        }
      for (i=2;i<=imax;i++)
        {
          i1=i-1;
          for (j=1;j<=imax;j++)
            BW[i][j]=0.0;
        }
```

```
if (nclust==1)
  {
    fprintf(lis," Méthode du lien simple \n");
    fprintf(lis," ---------------------- \n \n");
    fprintf(lis,"Ngps   Height  Leaders       C-H        D-H
                  C-index     Gamma      Beale \n \n");
  };

if (nclust==2)
  {
    fprintf(lis," Méthode du centroïde \n");
    fprintf(lis," ---------------------- \n \n");
    fprintf(lis,"Ngps   Height  Leaders       C-H        D-H
                  C-index     Gamma      Beale \n \n");
  };
if (nclust==3)
  {
    fprintf(lis," Méthode de Ward \n");
    fprintf(lis," ---------------- \n \n");
    fprintf(lis,"Ngps   Height  Leaders       C-H        D-H
                  C-index     Gamma      Beale \n \n");
  };
if (nclust==4)
  {
    fprintf(lis," Méthode du lien complet \n");
    fprintf(lis," ---------------------- \n \n");
    fprintf(lis,"Ngps   Height  Leaders       C-H        D-H
                  C-index     Gamma      Beale \n \n");
  };

int k=0;
ttot=0.0; // Initialisation de l'inertie totale
for (i=1;i<=imax-1;i++)
  {
    for (j=i+1;j<=imax;j++)
      {
        k=k+1;
        DLO[k]=DT[j][i];
        ttot=ttot+DT[j][i];
      }
  }

ttot=ttot/double(imax); // Inertie totale

NC2=imax*(imax-1)/2;

badsrt(DLO,DHI,NC2); // Tri du tableau des distances en ordre croissant

N1=imax-1;
k=0;
```

```
ch=0.0;
wtot=0.0;
dtot=0.0;
ndtot=0;
hold=0.0;
btot=0.0;
cind=0.0;

for (i=1;i<=imax;i++)
  {
    P[i]=i; // Au départ, chaque élément forme sa propre classe
    Q[i]=1; // Au départ, chaque classe contient un seul élément
  };

// On boucle sur le nombre de classes à considérer (de N-1 à 2 classes)

L10:

k=k+1;
dmax=1.0e20; // Initialisation de la hauteur de la hiérarchie
for (i=1;i<=N1;i++)
  {
    if (P[i]==i)
      {
        int i1=i+1;
        for (j=i1;j<=imax;j++)
          {
            if (P[j]==j)
              {
                t=DT[i][j];
                if (t<=dmax)
                  {
                    ic=i;
                    jc=j;
                    dmax=t;
                  };
              };
          };
      };
  };

dw1=calcul(DT,P,Q,imax,ic);
ssq1=dw1/double(Q[ic]); // Inertie intra-classes 1

dw2=calcul(DT,P,Q,imax,jc);
ssq2=dw2/double(Q[jc]); // Inertie intra-classes 2

// Somme des dissimilarités d'éléments appartenant aux classes ic et jc
dwt=dw1+dw2;
```

```
// Calcul de variables liées au C-index
nw1=(Q[ic]*(Q[ic]-1))/2;
nw2=(Q[jc]*(Q[jc]-1))/2;
nwt=nw1+nw2;

ssqt=ssq1+ssq2; // Somme des deux inerties intra-classes

for (j=1;j<=imax;j++)
  {
    if (P[j]==jc) // Si j se trouve dans la classe jc
      {
        for (i=1;i<=imax;i++)
          {
            if (P[i]==ic) // Si i se trouve dans la classe ic
              {
                ii=maxint(i,j);
                jj=minint(i,j);
                BW[ii][jj]=1.0;
              };
            P[j]=ic; // On met j dans la classe ic
          };
      };
  };

A[k]=ic;
B[k]=jc;
H[k]=dmax; // Distance entre les deux éléments que l'on regroupe

qic=Q[ic]; // Nombre d'éléments dans la classe ic
qjc=Q[jc]; // Nombre d'éléments dans la classe jc

z=qic+qjc; // Nombre total d'éléments dans les deux classes regroupées
ff=1.0/double(z);

for (i=1;i<=imax;i++) // Boucle sur les individus
  {
    if ((i!=ic)&&(P[i]==i))
      {
        if (nclust==3)
          {
            qi=Q[i];
            z=qi+qic+qjc;
            ff=1.0/double(z);
          }
        j=minint(ic,i);
        l=maxint(ic,i);
        k1=minint(jc,i);
        k2=maxint(jc,i);
        dj=DT[j][l];
        dk=DT[k1][k2];
```

```
        if (nclust==1) // Lien simple
          {
            DT[j][l]=mindou(dj,dk);
          }
        if (nclust==2) // Centroïde
          {
            DT[j][l]=ff*((double(qic)*dj)+(double(qjc)*dk));
          }
        if (nclust==3) // Ward
          {
            DT[j][l]=ff*((double(qic+qi)*dj)+(double(qjc+qi)*dk)-(qi*dmax));
          }
        if (nclust==4) // Lien complet
          {
            DT[j][l]=maxdou(dj,dk);
          };
      };
  }; // Fin de la boucle sur les individus

Q[ic]=Q[ic]+Q[jc]; // Mise à jour du nombre d'individus dans la classe ic

dw=calcul(DT,P,Q,imax,ic); // Somme des dissimilarités des
                           // éléments appartenant à la nouvelle classe ic
ssq=dw/(Q[ic]); // Mise à jour de l'inertie intra-classes

nqc=Q[ic];
nw=(nqc*(nqc-1))/2;

fq=double(nqc);
fp=double(nvar);

// Calcul de l'indice de Beale
if (ssqt-eps > 0.0)
  {
    beale=(ssq-ssqt)/ssqt;
    p1=2.0/fp;
    denom=(((fq-1.0)/(fq-2.0))*(pow(2.0,p1)))-1.0;
    beale=beale/denom;
  }
else beale=0.0;

// Calcul de l'indice de Duda-Heart

hold=((ssqt/ssq)-1.0+(2.0/(pi*fp)));
hold2=(2.0-16.0/(pi*pi*fp))/(fp*fq);
hold=-hold/sqrt(hold2);

// Calcul de la variance intra- et inter-classes
```

```
wtot=wtot+ssq-ssqt;
btot=ttot-wtot;

if (k!=N1) // S'il reste au-moins deux classes à considérer
  {
    // Calcul de l'indice de Calinski-Harabasz
    z=imax-k-1;
    ch=(btot*double(k))/(wtot*double(z));

    // Calcul de l'indice du C-index
    dtot=dtot+dw-dwt;
    ndtot=ndtot+nw-nwt;
    cind=(dtot-DLO[ndtot])/DHI[ndtot];

    // Calcul de l'indice Gamma
    kin=0;
    kout=0;

    for (i=2;i<=imax;i++)
      {
        i1=i-1;
        for (j=1;j<=i1;j++)
          {
            if (BW[i][j]<=0.0)
              {
                kout=kout+1;
                BB[kout]=DT[i][j];
              }
            else
              {
                kin=kin+1;
                W[kin]=DT[i][j];
              };
          };
      };

    u=0.0;
    dd=0.0;

    for (kk=1;kk<=kin;kk++)
      {
        din=W[kk];
        for (ll=1;ll<=kout;ll++)
          {
            if ((din-BB[ll]) ==0) dd=dd+0.5;
            if ((din-BB[ll]) > 0) u=u+1.0;
          };
      };

    denmr=(double(kin)*double(kout)/2.0)-dd;
```

```
            gamma=(1.0-u/denmr);

            // Décrémentation du nombre de classes à considérer
            ngp=imax-k;

            // Ecriture des résultats dans le fichier
            fprintf(lis," %2d %9.3f  %2d  %2d
                    %10.5f   %8.5f   %8.5f   %8.5f   %8.5f  \n",
                    ngp,H[k],A[k],B[k],ch,hold,cind,gamma,beale);
            if (ngp==kmax)
              {
                fprintf(lis,"Partition ");
                for (i=1;i<=imax;i++) fprintf(lis,"%3d",P[i]);
                fprintf(lis,"\n");
              }
            // On boucle sur le nombre de classes

            goto L10;

          };

        ngp=imax-k; // c.p. Il reste à calculer les indices pour le cas où on a 1
        // classe (Duda-Heart et Beale)

        fprintf(lis," %2d %9.3f  %2d  %2d         -
                %8.5f        -          -      %8.5f \n \n",
                ngp,H[k],A[k],B[k],hold,beale);

    }; // Fin de la boucle sur la méthode de classification


  for (ii = 0;ii<=imax;ii++)
    delete[imax+1] DT[ii];
  for (ii = 0;ii<=imax;ii++)
    delete[imax+1] BW[ii];
  delete DLO;
  delete DHI;
  delete P;
  delete Q;
  delete A;
  delete B;
  delete H;
};
```

Annexe B

Listing du fichier classes2.h

```
#ifndef __CLASSES2_H__
#define __CLASSES2_H__

void codage(FILE *lis,scluster para,CompSymbMat *M,int vsel[],
                                    double **x,double **d);
void milligan_scluster_m(FILE *lis,scluster para,double **d,int **t_classe);

void milligan_m(FILE *lis,scluster para,double **d);

#endif
```

Annexe C

Listing du fichier classes.cpp

Seules les fonctions utilisées par le fichier classes2.cpp sont présentées ci-dessous.

```
/* ================================================================== */
int badsrt(double *dlo,double *dhi,int nc2)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Ordonner par valeurs croissantes le tableau de distances DLO       */
/*                                                                    */
/* ================================================================== */
{
  int i,j;
  int ok2=1;
  double vec=0.;

  for (i=1;i<=nc2-1;i++)
    {
      vec=dlo[i];
      for (j=i+1;j<=nc2;j++)
        {
          if (vec-dlo[j] > 0)
            {
              dlo[i]=dlo[j];
              dlo[j]=vec;
              vec=dlo[i];
            };
        };
    };
```

```
  dhi[1]=dlo[nc2];

  for (i=2;i<=nc2;i++) dhi[i]=dhi[i-1]+dlo[nc2-i+1];
  for (i=2;i<=nc2;i++) dlo[i]=dlo[i-1]+dlo[i];
  for (i=1;i<=nc2;i++) dhi[i]=dhi[i]-dlo[i];

  return ok2;

};


/* ================================================================ */
double calcul(double **d,int *p,int *q,int n,int ic)
/* ================================================================ */
/*                                                                  */
/* Objectif :                                                       */
/*                                                                  */
/* Calcul de la somme "dw" et la moyenne "ssq" des dissimilarités   */
/* des éléments appartenant a la même classe que "ic".              */
/*                                                                  */
/* ================================================================ */
{
  int i,j;
  double ssq=0.0;
  double dw=0.0;

  if (q[ic]>1)
    {
     for (i=1;i<=n-1;i++)
       {
        if (p[i]==ic)
          {
            for (j=i+1;j<=n;j++)
              {
                if (p[j]==ic)
                  {
                    ssq=ssq + d[j][i];
                  };
              };
          };
       };
    };

  dw=ssq;
  ssq=ssq/double(q[ic]);
  return dw;
};
```

```
/* ================================================================== */
int minint(int a, int b)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Déterminer le minimum entre 2 valeurs entières                     */
/*                                                                    */
/* ================================================================== */
{
  int m1;
  m1 = (a < b) ? a : b;
  return m1;
};


/* ================================================================== */
int maxint(int a, int b)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Déterminer le maximum entre 2 valeurs entières                     */
/*                                                                    */
/* ================================================================== */
{
  int m2;
  m2 = (a < b) ? b : a;
  return m2;
};


/* ================================================================== */
double mindou(double a, double b)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Déterminer le minimum entre 2 valeurs réelles                      */
/*                                                                    */
/* ================================================================== */
{
  double m3;
  m3 = (a < b) ? a : b;
  return m3;
};
```

```
/* ================================================================== */
double maxdou(double a, double b)
/* ================================================================== */
/*                                                                    */
/* Objectif :                                                         */
/*                                                                    */
/* Déterminer le maximum entre 2 valeurs réelles                      */
/*                                                                    */
/* ================================================================== */
{
  double m4;
  m4 = (a < b) ? b : a;
  return m4;
};
```

Annexe D

Jeu de données merovingian.sds

	fixation	Bordures	damasquinure	Fond	incrustation	plaque
01	clous	bords, repet	bichr, placa	plaqu	filif	arabe, grand
02	clous	bords, repet	bichr, placa	plaqu	filif, bande	dorsa, motif
03	bosse	frise	inscr, monoc	Missing Value	bande	tress
04	clous	frise	inscr, monoc	hachu	bande	dorsa, tress
05	bosse	frise	inscr, monoc	trame	Missing Value	Missing Value
06	clous	bords, repet	bichr, placa	plaqu	filif	dorsa, motif
07	bosse	bords	inscr, monoc	hachu	bande	dorsa, motif
08	No	bords	bichr, placa	plaqu	bande	motif
09	bosse	frise	inscr, monoc	hachu	bande	dorsa, motif, tress
10	clous	repet	bichr, placa	plaqu	filif	arabe, grand
11	bosse	Missing Value	inscr, monoc	hachu	ruban	dorsa
12	clous	bords, repet	bichr, placa	plaqu	filif	motif

	fixation	Bordures	damasquinure	Fond	incrustation	plaque
13	clous	repet	bichr, placa	plaqu	filif	dorsa, motif
14	bosse	frise	inscr, monoc	hachu	bande	Missing Value
15	bosse	Missing Value	inscr, monoc	trame	Missing Value	ronde
16	clous	bords, repet	bichr, placa	plaqu	filif	arabe, grand
17	bosse	Missing Value	inscr, monoc	hachu	ruban	Missing Value
18	clous	bords, repet	bichr, placa	plaqu	filif	motif
19	clous	frise	bichr, placa	plaqu	filif, bande	motif
20	clous	bords, frise	inscr, monoc	Missing Value	filif	tress
21	clous	frise	inscr, monoc	hachu	bande	grand, tress
22	clous	frise	inscr, monoc	hachu	bande	dorsa, tress
23	bosse	Missing Value	inscr, monoc	trame	Missing Value	ronde
24	bosse	frise	inscr, monoc	hachu	bande	dorsa, tress

	fixation	Bordures	damasquinure	Fond	incrustation	plaque
25	bosse	frise	inscr, monoc	hachu	bande	dorsa, tress
26	bosse	frise	inscr, monoc	hachu	bande	grand, tress
27	bosse	Missing Value	inscr, monoc	hachu	ruban	Missing Value
28	bosse	frise	inscr, monoc	hachu	bande, bande	grand, tress
29	clous	frise	bichr, placa	plaqu	bande	dorsa, motif
30	bosse	frise	inscr, monoc	hachu	bande	tress
31	bosse	bords, frise	inscr, monoc	hachu	bande	tress
32	bosse	frise	inscr, monoc	hachu	bande	dorsa, tress
33	bosse	frise	inscr, monoc	hachu	bande	dorsa, tress
34	bosse	frise	inscr, monoc	hachu	ruban	dorsa
35	bosse	frise	inscr, monoc	plaqu	bande	tress
36	bosse	frise	inscr, monoc	hachu	bande	tress

	fixation	Bordures	damasquinure	Fond	incrustation	plaque
37	clous	Missing Value	bichr, placa	plaqu	filif	arabe, grand
38	clous	bords	bichr, placa	plaqu	filif	arabe, grand
39	clous	bords	bichr, placa	plaqu	filif	arabe, grand
40	clous	bords	bichr, placa	plaqu	filif	arabe
41	clous	bords	bichr, placa	plaqu	filif	arabe
42	clous	bords	bichr, placa	plaqu	filif	arabe
43	clous	bords	bichr, placa	plaqu	filif	arabe
44	clous	bords	bichr, placa	plaqu	filif	motif
45	clous	bords	bichr, placa	plaqu	filif	tress
46	bosse	frise	inscr, monoc	hachu	bande	dorsa, tress
47	bosse	Missing Value	inscr, monoc	trame	Missing Value	ronde
48	bosse	bords	inscr, monoc	Missing Value	ruban	Missing Value

	fixation	Bordures	damasquinure	Fond	incrustation	plaque
49	bosse	frise	inscr, monoc	hachu	bande	dorsa, tress
50	bosse	bords	inscr, monoc	hachu	bande	dorsa, motif
51	bosse	Missing Value	inscr	hachu	ruban	Missing Value
52	bosse	frise	inscr, monoc	hachu	bande	dorsa
53	bosse	frise	inscr, monoc	hachu	bande	dorsa, tress
54	bosse	Missing Value	inscr, monoc	trame	Missing Value	ronde
55	bosse	Missing Value	inscr, monoc	trame	Missing Value	ronde
56	bosse	Missing Value	inscr, monoc	trame	Missing Value	ronde
57	bosse	Missing Value	inscr, monoc	trame	Missing Value	ronde
58	bosse	Missing Value	inscr, monoc	trame	Missing Value	ronde
59	bosse	Missing Value	inscr, monoc	hachu	ruban	Missing Value

Annexe E

Jeu de données animaux.sds

	mode_vie	mode_deplacemen	race	mode_reproducti	mode_respirato	type_peau	milieu_vie	alimentation	couve_allaite_e
aigle	sauva	elles, patte	impér	ovipa	poumo	plume	air	carni	couve
chat	domes	patte	persa	vivip	poumo	poils	sol	omniv	allai
cheval	domes	patte	pur s	vivip	poumo	poils	sol	herbi	allai
chien	sauva	patte	labra	vivip	poumo	poils	sol	carni	allai
hirondelle	sauva	ailes, patte	rusti	ovipa	poumo	plume	air	insec	couve
oie	sauva	ailes, palme	blanc	ovipa	poumo	plume	air, sol	herbi	couve
ours	sauva	patte	brun	vivip	poumo	poils	sol	omniv	allai
pigeon	sauva	ailes, patte	texan	ovipa	poumo	plume	air	grani	couve
poule	éleva	ailes, patte	naine	ovipa	poumo	plume	sol	grani	couve
requin	sauva	nageo	chat	ovipa	branc	écail	eau	carni	aucun
saumon	sauva	nageo	atlan	ovipa	branc	écail	eau	carni	aucun
thon	sauva	nageo	rouge	ovipa	branc	écail	eau	carni	aucun
truite	éleva	nageo	arc-e	ovipa	branc	écail	eau	carni	aucun
vache	éleva	patte	norma	vivip	poumo	poils	sol	herbi	allai

Annexe F

Jeu de données conso.sds

	Type of first d
North non-metro	Separ (0.09), Washi (0.21), CD pl (0.09), Micro (0.14), Separ (0.08), Tumbl (0.10), Video (0.16), Dish (0.03), Fridg (0.10)
Yorks and humbe	Separ (0.10), Washi (0.17), CD pl (0.08), Micro (0.18), Separ (0.09), Tumbl (0.11), Video (0.15), Dish (0.02), Fridg (0.12)
Yorks and humbe	Separ (0.11), Washi (0.17), CD pl (0.08), Micro (0.12), Separ (0.09), Tumbl (0.10), Video (0.18), Dish (0.03), Fridg (0.12)
East midlands n	Separ (0.10), Washi (0.21), CD pl (0.08), Micro (0.13), Separ (0.08), Tumbl (0.11), Video (0.15), Dish (0.04), Fridg (0.09), None (0.00)
North west metr	Separ (0.09), Washi (0.20), CD pl (0.08), Micro (0.14), Separ (0.06), Tumbl (0.09), Video (0.16), Dish (0.03), Fridg (0.15)
North west non-	Separ (0.10), Washi (0.17), CD pl (0.10), Micro (0.14), Separ (0.07), Tumbl (0.10), Video (0.16), Dish (0.02), Fridg (0.14)
South east othe	Separ (0.12), Washi (0.15), CD pl (0.10), Micro (0.13), Separ (0.10), Tumbl (0.09), Video (0.14), Dish (0.05), Fridg (0.11)
West midlands m	Separ (0.14), Washi (0.18), CD pl (0.10), Micro (0.12), Separ (0.07), Tumbl (0.08), Video (0.17), Dish (0.03), Fridg (0.12), None (0.00)
West midlands n	Separ (0.12), Washi (0.15), CD pl (0.09), Micro (0.13), Separ (0.11), Tumbl (0.12), Video (0.17), Dish (0.03), Fridg (0.08)
East anglia	Separ (0.13), Washi (0.17), CD pl (0.07), Micro (0.12), Separ (0.11), Tumbl (0.12), Video (0.14), Dish (0.05), Fridg (0.08)
Greater london	Separ (0.09), Washi (0.20), CD pl (0.09), Micro (0.11), Separ (0.08), Tumbl (0.09), Video (0.15), Dish (0.04), Fridg (0.15), None (0.00)
Greater london	Separ (0.10), Washi (0.17), CD pl (0.09), Micro (0.10), Separ (0.08), Tumbl (0.09), Video (0.19), Dish (0.05), Fridg (0.14)
Greater london	Separ (0.09), Washi (0.19), CD pl (0.09), Micro (0.12), Separ (0.09), Tumbl (0.10), Video (0.14), Dish (0.04), Fridg (0.13)
Greater london	Separ (0.09), Washi (0.17), CD pl (0.12), Separ (0.06), Tumbl (0.09), Video (0.17), Dish (0.05), Fridg (0.14), None (0.00)
South east metr	Separ (0.09), Washi (0.16), CD pl (0.09), Micro (0.14), Separ (0.11), Tumbl (0.13), Video (0.15), Dish (0.04), Fridg (0.11)
South west	Separ (0.11), Washi (0.17), CD pl (0.09), Micro (0.13), Separ (0.10), Tumbl (0.11), Video (0.13), Dish (0.04), Fridg (0.12)
Wales / Gwent	Separ (0.09), Washi (0.20), CD pl (0.08), Micro (0.16), Separ (0.09), Tumbl (0.11), Video (0.14), Dish (0.02), Fridg (0.11), None (0.00)
Wales ii (otw	Separ (0.11), Washi (0.19), CD pl (0.05), Micro (0.16), Separ (0.10), Tumbl (0.10), Video (0.13), Dish (0.04), Fridg (0.13)
Scotland i (gra	Separ (0.09), Washi (0.20), CD pl (0.10), Micro (0.15), Separ (0.07), Tumbl (0.11), Video (0.14), Dish (0.02), Fridg (0.11)
Scotland ii (lo	Separ (0.09), Washi (0.21), CD pl (0.09), Micro (0.12), Separ (0.07), Tumbl (0.14), Video (0.16), Dish (0.04), Fridg (0.11)
Scotland iii (mo	Separ (0.09), Washi (0.19), CD pl (0.09), Micro (0.13), Separ (0.05), Tumbl (0.13), Video (0.14), Dish (0.04), Fridg (0.14)
Scotland iv (no	Separ (0.11), Washi (0.19), CD pl (0.09), Micro (0.14), Separ (0.05), Tumbl (0.12), Video (0.14), Dish (0.04), Fridg (0.11), None (0.00)
Scotland v (du	Separ (0.15), Washi (0.21), CD pl (0.08), Micro (0.12), Separ (0.09), Tumbl (0.10), Video (0.15), Dish (0.03), Fridg (0.07)
Northern irelan	Separ (0.10), Washi (0.20), CD pl (0.09), Micro (0.14), Separ (0.04), Tumbl (0.08), Video (0.17), Dish (0.03), Fridg (0.14)

	GMVEserwerages-r	GMVEtv-licence	GMVEwater
North non-metro	v=0 (1.00)	v2-3 (0.90), v0-1 (0.10)	v=0 (0.36), v1-2 (0.10), v=3 (0.19), v=4 (0.22), v=5 (0.06), v>5 (0.06)
Yorks and humbe	v=0 (1.00)	v2-3 (0.90), v0-1 (0.10)	v=0 (0.31), v1-2 (0.12), v=3 (0.16), v=4 (0.22), v=5 (0.11), v>5 (0.08)
Yorks and humbe	v=0 (0.87), v=1 (0.04), v=2 (0.06), v=3 (0.02), v>3 (0.01)	v2-3 (0.96), v0-1 (0.04)	v=0 (0.25), v1-2 (0.19), v=3 (0.19), v=4 (0.17), v=5 (0.10), v>5 (0.09)
East midlands n	v=0 (0.97), v=1 (0.01), v=2 (0.01), v=3 (0.00)	v2-3 (0.92), v0-1 (0.08)	v=0 (0.33), v1-2 (0.12), v=3 (0.14), v=4 (0.16), v=5 (0.14), v>5 (0.12)
North west metr	v=0 (1.00)	v2-3 (0.84), v0-1 (0.16)	v=0 (0.38), v1-2 (0.12), v=3 (0.12), v=4 (0.16), v=5 (0.12), v>5 (0.10)
North west non-	v=0 (1.00)	v2-3 (0.92), v0-1 (0.08)	v=0 (0.24), v1-2 (0.18), v=3 (0.22), v=4 (0.22), v=5 (0.06), v>5 (0.07)
South east othe	v=0 (0.74), v=1 (0.04), v=2 (0.11), v=3 (0.08), v>3 (0.02)	v2-3 (0.89), v0-1 (0.11)	v=0 (0.29), v1-2 (0.24), v=3 (0.15), v=4 (0.14), v=5 (0.08), v>5 (0.09)
West midlands m	v=0 (1.00), v=1 (0.00)	v2-3 (0.83), v0-1 (0.17)	v=0 (0.40), v1-2 (0.08), v=3 (0.18), v=4 (0.18), v=5 (0.10), v>5 (0.06)
West midlands n	v=0 (0.99), v=1 (0.01), v=2 (0.00), v=3 (0.00)	v2-3 (0.94), v0-1 (0.06)	v=0 (0.25), v1-2 (0.14), v=3 (0.19), v=4 (0.20), v=5 (0.14), v>5 (0.09)
East anglia	v=0 (0.90), v=1 (0.00), v=2 (0.05), v=3 (0.05), v>3 (0.00)	v2-3 (0.86), v0-1 (0.14)	v=0 (0.33), v1-2 (0.12), v=3 (0.08), v=4 (0.15), v=5 (0.14), v>5 (0.18)
Greater london	v=0 (1.00)	v2-3 (0.84), v0-1 (0.16)	v=0 (0.41), v1-2 (0.15), v=3 (0.24), v=4 (0.13), v=5 (0.06), v>5 (0.02)
Greater london	v=0 (1.00)	v2-3 (0.84), v0-1 (0.16)	v=0 (0.39), v1-2 (0.09), v=3 (0.19), v=4 (0.20), v=5 (0.08), v>5 (0.05)
Greater london	v=0 (1.00)	v2-3 (0.83), v0-1 (0.17)	v=0 (0.48), v1-2 (0.08), v=3 (0.25), v=4 (0.08), v=5 (0.05), v>5 (0.05)
Greater london	v=0 (0.92), v=1 (0.05), v=2 (0.03)	v2-3 (0.94), v0-1 (0.06)	v=0 (0.29), v1-2 (0.12), v=3 (0.29), v=4 (0.19), v=5 (0.04), v>5 (0.06)
South east metr	v=0 (0.93), v=1 (0.02), v=2 (0.03), v=3 (0.01), v>3 (0.00)	v2-3 (0.92), v0-1 (0.08)	v=0 (0.24), v1-2 (0.13), v=3 (0.16), v=4 (0.18), v=5 (0.14), v>5 (0.15)
South west	v=0 (0.77), v=1 (0.04), v=2 (0.09), v=3 (0.08), v>3 (0.01)	v2-3 (0.92), v0-1 (0.08)	v=0 (0.24), v1-2 (0.26), v=3 (0.16), v=4 (0.11), v=5 (0.09), v>5 (0.13)
Wales I (gwent	v=0 (1.00)	v2-3 (0.89), v0-1 (0.11)	v=0 (0.34), v1-2 (0.01), v=3 (0.03), v=4 (0.22), v=5 (0.22), v>5 (0.17)
Wales II (clw	v=0 (1.00)	v2-3 (0.90), v0-1 (0.10)	v=0 (0.42), v1-2 (0.10), v=3 (0.05), v=4 (0.17), v=5 (0.19), v>5 (0.08)
Scotland I (gra	v=0 (1.00)	v2-3 (0.91), v0-1 (0.09)	v=0 (1.00)
Scotland II (lo	v=0 (1.00)	v2-3 (0.91), v0-1 (0.09)	v=0 (1.00)
Scotland III me	v=0 (1.00)	v2-3 (0.81), v0-1 (0.19)	v=0 (1.00)
Scotland III no	v=0 (1.00)	v2-3 (0.92), v0-1 (0.08)	v=0 (1.00)
Scotland IV (du	v=0 (1.00)	v2-3 (0.94), v0-1 (0.06)	v=0 (1.00)
Northern Irelan	v=0 (1.00)	v2-3 (0.77), v0-1 (0.23)	v=0 (1.00)

	Number of adult	Central heating	Fuel type centr	Central heating	CH repairs last
North non-metro	[1.00 : 6.00]	No (0.12), Yes (0.88)	Mains (0.71), Solid (0.11), Elect (0.10), Oil (0.07), Bottl (0.00), Solid (0.00)	Yes (0.02), No (0.98)	No (0.73), Yes (0.27)
Yorks and humbe	[1.00 : 6.00]	No (0.23), Yes (0.77)	Mains (0.83), Solid (0.08), Elect (0.09), Oil (0.00), Bottl (0.00)	Yes (0.04), No (0.96)	No (0.69), Yes (0.31)
Yorks and humbe	[1.00 : 4.00]	No (0.19), Yes (0.81)	Mains (0.76), Solid (0.06), Elect (0.11), Oil (0.06), Solid (0.01), Other (0.01)	Yes (0.03), No (0.97)	No (0.69), Yes (0.31)
East midlands n	[1.00 : 6.00]	No (0.12), Yes (0.88)	Mains (0.78), Solid (0.06), Elect (0.09), Oil (0.04), Bottl (0.01), Solid (0.00), Other (0.00)	Yes (0.00), No (1.00)	No (0.71), Yes (0.29)
North west metr	[1.00 : 6.00]	No (0.22), Yes (0.78)	Mains (0.90), Solid (0.01), Elect (0.08), Oil (0.01), Solid (0.00)	Yes (0.01), No (0.99)	No (0.75), Yes (0.25)
North west non-	[1.00 : 4.00]	No (0.22), Yes (0.78)	Mains (0.87), Solid (0.01), Elect (0.10), Oil (0.01), Bottl (0.00), Other (0.00)	Yes (0.04), No (0.96)	No (0.74), Yes (0.26)
South east othe	[1.00 : 6.00]	No (0.13), Yes (0.87)	Mains (0.78), Solid (0.02), Elect (0.13), Oil (0.05), Bottl (0.00), Solid (0.01), Other (0.00)	Yes (0.02), No (0.98)	No (0.69), Yes (0.31)
West midlands m	[1.00 : 5.00]	No (0.24), Yes (0.76)	Mains (0.91), Elect (0.09)	Yes (0.01), No (0.99)	No (0.73), Yes (0.27)
West midlands n	[1.00 : 4.00]	No (0.14), Yes (0.86)	Mains (0.73), Solid (0.08), Elect (0.11), Oil (0.05), Bottl (0.02), Other (0.01)	Yes (0.02), No (0.98)	No (0.73), Yes (0.27)
East anglia	[1.00 : 5.00]	No (0.10), Yes (0.90)	Mains (0.59), Solid (0.07), Elect (0.17), Oil (0.14), Bottl (0.01), Other (0.02)	Yes (0.03), No (0.97)	No (0.76), Yes (0.24)
Greater london	[1.00 : 5.00]	No (0.19), Yes (0.81)	Mains (0.90), Solid (0.01), Elect (0.08), Other (0.02)	Yes (0.03), No (0.98)	No (0.70), Yes (0.30)
Greater london	[1.00 : 5.00]	No (0.09), Yes (0.91)	Mains (0.84), Elect (0.14), Oil (0.01)	Yes (0.04), No (0.96)	No (0.76), Yes (0.24)
Greater london	[1.00 : 4.00]	No (0.17), Yes (0.83)	Mains (0.82), Elect (0.11), Oil (0.01), Other (0.06)	Yes (0.02), No (0.98)	No (0.70), Yes (0.30)
Greater london	[1.00 : 5.00]	No (0.12), Yes (0.88)	Mains (0.88), Elect (0.09), Oil (0.02), Other (0.01)	No (1.00)	No (0.67), Yes (0.33)
South east metr	[1.00 : 5.00]	No (0.09), Yes (0.91)	Mains (0.85), Solid (0.01), Elect (0.10), Oil (0.04), Bottl (0.01), Other (0.00)	Yes (0.01), No (0.99)	No (0.66), Yes (0.34)
South west	[1.00 : 5.00]	No (0.16), Yes (0.84)	Mains (0.71), Solid (0.03), Elect (0.17), Oil (0.07), Bottl (0.01), Solid (0.00), Other (0.00)	Yes (0.02), No (0.98)	No (0.68), Yes (0.32)
Wales I (gwent	[1.00 : 4.00]	No (0.13), Yes (0.87)	Mains (0.87), Solid (0.09), Elect (0.04), Oil (0.01)	Yes (0.01), No (0.99)	No (0.82), Yes (0.18)
Wales ii (clw.	[1.00 : 5.00]	No (0.25), Yes (0.75)	Mains (0.32), Solid (0.24), Elect (0.24), Oil (0.13), Bottl (0.05), Other (0.01)	Yes (0.02), No (0.98)	No (0.84), Yes (0.16)
Scotland i (gra	[1.00 : 4.00]	No (0.14), Yes (0.86)	Mains (0.50), Solid (0.12), Elect (0.28), Oil (0.08), Bottl (0.03)	Yes (0.01), No (0.99)	No (0.73), Yes (0.27)
Scotland ii (lo	[1.00 : 4.00]	No (0.12), Yes (0.88)	Mains (0.69), Solid (0.13), Elect (0.18), Bottl (0.01)	Yes (0.01), No (0.99)	No (0.77), Yes (0.23)
Scotland iii me	[1.00 : 5.00]	No (0.21), Yes (0.79)	Mains (0.79), Solid (0.01), Elect (0.20)	Yes (0.06), No (0.94)	No (0.78), Yes (0.22)
Scotland iii no	[1.00 : 4.00]	No (0.07), Yes (0.93)	Mains (0.72), Solid (0.05), Elect (0.19), Oil (0.01), Bottl (0.01), Solid (0.01), Other (0.01)	Yes (0.02), No (0.98)	No (0.75), Yes (0.25)
Scotland iv (du	[1.00 : 4.00]	No (0.28), Yes (0.72)	Mains (0.43), Elect (0.43), Oil (0.09), Bottl (0.04)	Yes (0.04), No (0.96)	No (0.87), Yes (0.13)
Northern irelan	[1.00 : 6.00]	No (0.09), Yes (0.91)	Solid (0.41), Elect (0.14), Oil (0.38), Bottl (0.01), Solid (0.04), Other (0.02)	Yes (0.04), No (0.96)	No (0.78), Yes (0.22)

	QWEfurniture	QWEoil	QWEcarpets
North non-metro	v=0 (0.82), v1-5 (0.04), v6-20 (0.06), v>20 (0.09)	v=0 (0.97), v1-10 (0.02), v>10 (0.02)	v=0 (0.87), v1-5 (0.04), v6-20 (0.06), v>20 (0.03)
Yorks and humbe	v=0 (0.81), v1-5 (0.06), v6-20 (0.07), v>20 (0.07)	v=0 (1.00), v>10 (0.00)	v=0 (0.85), v1-5 (0.06), v6-20 (0.06), v>20 (0.03)
Yorks and humbe	v=0 (0.80), v1-5 (0.08), v6-20 (0.05), v>20 (0.07)	v=0 (0.96), v1-10 (0.01), v>10 (0.03)	v=0 (0.82), v1-5 (0.09), v6-20 (0.04), v>20 (0.05)
East midlands n	v=0 (0.81), v1-5 (0.06), v6-20 (0.08), v>20 (0.05)	v=0 (0.98), v1-10 (0.01), v>10 (0.01)	v=0 (0.85), v1-5 (0.07), v6-20 (0.05), v>20 (0.03)
North west metr	v=0 (0.82), v1-5 (0.05), v6-20 (0.06), v>20 (0.07)	v=0 (1.00), v>10 (0.00)	v=0 (0.85), v1-5 (0.06), v6-20 (0.07), v>20 (0.02)
North west non-	v=0 (0.81), v1-5 (0.05), v6-20 (0.07), v>20 (0.07)	v=0 (0.99), v1-10 (0.00), v>10 (0.01)	v=0 (0.89), v1-5 (0.05), v6-20 (0.03), v>20 (0.03)
South east othe	v=0 (0.79), v1-5 (0.07), v6-20 (0.07), v>20 (0.07)	v=0 (0.96), v1-10 (0.02), v>10 (0.02)	v=0 (0.88), v1-5 (0.05), v6-20 (0.03), v>20 (0.03)
West midlands m	v=0 (0.81), v1-5 (0.06), v6-20 (0.05), v>20 (0.08)	v=0 (1.00)	v=0 (0.85), v1-5 (0.07), v6-20 (0.05), v>20 (0.03)
West midlands n	v=0 (0.81), v1-5 (0.06), v6-20 (0.05), v>20 (0.07)	v=0 (0.97), v1-10 (0.01), v>10 (0.02)	v=0 (0.86), v1-5 (0.05), v6-20 (0.05), v>20 (0.03)
East anglia	v=0 (0.80), v1-5 (0.06), v6-20 (0.07), v>20 (0.07)	v=0 (0.92), v1-10 (0.05), v>10 (0.03)	v=0 (0.89), v1-5 (0.05), v6-20 (0.03), v>20 (0.03)
Greater london	v=0 (0.87), v1-5 (0.01), v6-20 (0.05), v>20 (0.07)	v=0 (1.00)	v=0 (0.88), v1-5 (0.05), v6-20 (0.04), v>20 (0.03)
Greater london	v=0 (0.85), v1-5 (0.04), v6-20 (0.03), v>20 (0.08)	v=0 (1.00)	v=0 (0.92), v1-5 (0.05), v6-20 (0.01), v>20 (0.03)
Greater london	v=0 (0.82), v1-5 (0.05), v6-20 (0.07), v>20 (0.06)	v=0 (1.00)	v=0 (0.87), v1-5 (0.08), v6-20 (0.03), v>20 (0.03)
Greater london	v=0 (0.80), v1-5 (0.06), v6-20 (0.06), v>20 (0.08)	v=0 (1.00)	v=0 (0.93), v1-5 (0.04), v6-20 (0.03), v>20 (0.01)
South east metr	v=0 (0.84), v1-5 (0.03), v6-20 (0.07), v>20 (0.06)	v=0 (0.98), v1-10 (0.01), v>10 (0.01)	v=0 (0.88), v1-5 (0.05), v6-20 (0.04), v>20 (0.03)
South west	v=0 (0.82), v1-5 (0.05), v6-20 (0.07), v>20 (0.06)	v=0 (0.97), v1-10 (0.01), v>10 (0.02)	v=0 (0.84), v1-5 (0.09), v6-20 (0.04), v>20 (0.03)
Wales I (gwent	v=0 (0.85), v1-5 (0.05), v6-20 (0.04), v>20 (0.05)	v=0 (1.00)	v=0 (0.90), v1-5 (0.05), v6-20 (0.03), v>20 (0.03)
Wales II (clw,	v=0 (0.86), v1-5 (0.03), v6-20 (0.05), v>20 (0.07)	v=0 (0.92), v1-10 (0.02), v>10 (0.06)	v=0 (0.87), v1-5 (0.05), v6-20 (0.07), v>20 (0.01)
Scotland I (gra	v=0 (0.78), v1-5 (0.06), v6-20 (0.09), v>20 (0.08)	v=0 (0.98), v1-10 (0.01), v>10 (0.01)	v=0 (0.81), v1-5 (0.06), v6-20 (0.08), v>20 (0.06)
Scotland II (lo	v=0 (0.81), v1-5 (0.06), v6-20 (0.08), v>20 (0.06)	v=0 (1.00)	v=0 (0.82), v1-5 (0.06), v6-20 (0.06), v>20 (0.06)
Scotland III me	v=0 (0.80), v1-5 (0.05), v6-20 (0.07), v>20 (0.07)	v=0 (1.00)	v=0 (0.84), v1-5 (0.07), v6-20 (0.06), v>20 (0.03)
Scotland III no	v=0 (0.82), v1-5 (0.06), v6-20 (0.08), v>20 (0.04)	v=0 (0.99), v1-10 (0.01), v>10 (0.01)	v=0 (0.88), v1-5 (0.06), v6-20 (0.02), v>20 (0.05)
Scotland IV (du	v=0 (0.84), v1-5 (0.06), v6-20 (0.03), v>20 (0.06)	v=0 (0.94), v1-10 (0.03), v>10 (0.03)	v=0 (0.94), v1-5 (0.03), v6-20 (0.03)
Northern irelan	v=0 (0.76), v1-5 (0.06), v6-20 (0.11), v>20 (0.08)	v=0 (0.70), v1-10 (0.17), v>10 (0.13)	v=0 (0.84), v1-5 (0.07), v6-20 (0.07), v>20 (0.02)

	QWEtelephone	QWEgas	QWEelect
North non-metro	v=0 (0.13), v1-5 (0.56), v6-10 (0.23), v>10 (0.08)	v=0 (0.32), v1-5 (0.18), v6-10 (0.20), v11-2 (0.23), v>20 (0.07)	v=0 (0.08), v1-5 (0.35), v6-10 (0.20), v11-2 (0.28), v>20 (0.09)
Yorks and humbe	v=0 (0.12), v1-5 (0.58), v6-10 (0.22), v>10 (0.09)	v=0 (0.17), v1-5 (0.28), v6-10 (0.20), v11-2 (0.28), v>20 (0.07)	v=0 (0.14), v1-5 (0.32), v6-10 (0.26), v11-2 (0.22), v>20 (0.07)
Yorks and humbe	v=0 (0.09), v1-5 (0.60), v6-10 (0.23), v>10 (0.07)	v=0 (0.24), v1-5 (0.25), v6-10 (0.22), v11-2 (0.20), v>20 (0.08)	v=0 (0.09), v1-5 (0.33), v6-10 (0.25), v11-2 (0.25), v>20 (0.09)
East midlands n	v=0 (0.11), v1-5 (0.59), v6-10 (0.22), v>10 (0.08)	v=0 (0.23), v1-5 (0.26), v6-10 (0.21), v11-2 (0.23), v>20 (0.07)	v=0 (0.14), v1-5 (0.34), v6-10 (0.26), v11-2 (0.21), v>20 (0.04)
North west metr	v=0 (0.14), v1-5 (0.54), v6-10 (0.23), v>10 (0.09)	v=0 (0.16), v1-5 (0.29), v6-10 (0.21), v11-2 (0.25), v>20 (0.09)	v=0 (0.16), v1-5 (0.32), v6-10 (0.28), v11-2 (0.19), v>20 (0.05)
North west non-	v=0 (0.09), v1-5 (0.60), v6-10 (0.22), v>10 (0.09)	v=0 (0.14), v1-5 (0.27), v6-10 (0.27), v11-2 (0.25), v>20 (0.06)	v=0 (0.10), v1-5 (0.32), v6-10 (0.28), v11-2 (0.22), v>20 (0.08)
South east othe	v=0 (0.10), v1-5 (0.59), v6-10 (0.22), v>10 (0.10)	v=0 (0.25), v1-5 (0.29), v6-10 (0.20), v11-2 (0.21), v>20 (0.06)	v=0 (0.13), v1-5 (0.30), v6-10 (0.31), v11-2 (0.21), v>20 (0.05)
West midlands m	v=0 (0.09), v1-5 (0.57), v6-10 (0.24), v>10 (0.10)	v=0 (0.13), v1-5 (0.30), v6-10 (0.23), v11-2 (0.28), v>20 (0.06)	v=0 (0.17), v1-5 (0.35), v6-10 (0.20), v11-2 (0.24), v>20 (0.04)
West midlands n	v=0 (0.12), v1-5 (0.56), v6-10 (0.24), v>10 (0.09)	v=0 (0.27), v1-5 (0.26), v6-10 (0.20), v11-2 (0.20), v>20 (0.06)	v=0 (0.10), v1-5 (0.38), v6-10 (0.26), v11-2 (0.18), v>20 (0.07)
East anglia	v=0 (0.12), v1-5 (0.58), v6-10 (0.23), v>10 (0.08)	v=0 (0.44), v1-5 (0.22), v6-10 (0.12), v11-2 (0.17), v>20 (0.04)	v=0 (0.12), v1-5 (0.42), v6-10 (0.22), v11-2 (0.20), v>20 (0.04)
Greater london	v=0 (0.14), v1-5 (0.47), v6-10 (0.24), v>10 (0.15)	v=0 (0.14), v1-5 (0.37), v6-10 (0.22), v11-2 (0.21), v>20 (0.06)	v=0 (0.20), v1-5 (0.36), v6-10 (0.21), v11-2 (0.20), v>20 (0.04)
Greater london	v=0 (0.08), v1-5 (0.44), v6-10 (0.26), v>10 (0.21)	v=0 (0.19), v1-5 (0.34), v6-10 (0.21), v11-2 (0.19), v>20 (0.07)	v=0 (0.10), v1-5 (0.44), v6-10 (0.28), v11-2 (0.14), v>20 (0.04)
Greater london	v=0 (0.13), v1-5 (0.42), v6-10 (0.32), v>10 (0.13)	v=0 (0.13), v1-5 (0.42), v6-10 (0.26), v11-2 (0.15), v>20 (0.04)	v=0 (0.17), v1-5 (0.32), v6-10 (0.30), v11-2 (0.18), v>20 (0.03)
Greater london	v=0 (0.04), v1-5 (0.64), v6-10 (0.19), v>10 (0.13)	v=0 (0.13), v1-5 (0.33), v6-10 (0.18), v11-2 (0.26), v>20 (0.06)	v=0 (0.13), v1-5 (0.36), v6-10 (0.30), v11-2 (0.18), v>20 (0.03)
South east metr	v=0 (0.06), v1-5 (0.51), v6-10 (0.29), v>10 (0.15)	v=0 (0.20), v1-5 (0.27), v6-10 (0.20), v11-2 (0.24), v>20 (0.10)	v=0 (0.10), v1-5 (0.33), v6-10 (0.28), v11-2 (0.22), v>20 (0.07)
South west	v=0 (0.08), v1-5 (0.59), v6-10 (0.23), v>10 (0.10)	v=0 (0.31), v1-5 (0.23), v6-10 (0.23), v11-2 (0.19), v>20 (0.04)	v=0 (0.13), v1-5 (0.29), v6-10 (0.32), v11-2 (0.21), v>20 (0.05)
Wales i (gwent	v=0 (0.14), v1-5 (0.58), v6-10 (0.19), v>10 (0.09)	v=0 (0.22), v1-5 (0.26), v6-10 (0.25), v11-2 (0.19), v>20 (0.05)	v=0 (0.19), v1-5 (0.35), v6-10 (0.26), v11-2 (0.17), v>20 (0.01)
Wales ii (clw,	v=0 (0.15), v1-5 (0.51), v6-10 (0.25), v>10 (0.09)	v=0 (0.71), v1-5 (0.10), v6-10 (0.10), v11-2 (0.07), v>20 (0.03)	v=0 (0.20), v1-5 (0.22), v6-10 (0.25), v11-2 (0.22), v>20 (0.10)
Scotland i (gra	v=0 (0.10), v1-5 (0.52), v6-10 (0.26), v>10 (0.12)	v=0 (0.53), v1-5 (0.11), v6-10 (0.11), v11-2 (0.17), v>20 (0.08)	v=0 (0.19), v1-5 (0.23), v6-10 (0.28), v11-2 (0.19), v>20 (0.12)
Scotland ii (lo	v=0 (0.10), v1-5 (0.54), v6-10 (0.24), v>10 (0.12)	v=0 (0.32), v1-5 (0.17), v6-10 (0.15), v11-2 (0.27), v>20 (0.08)	v=0 (0.21), v1-5 (0.26), v6-10 (0.27), v11-2 (0.19), v>20 (0.08)
Scotland iii me	v=0 (0.15), v1-5 (0.50), v6-10 (0.28), v>10 (0.07)	v=0 (0.25), v1-5 (0.21), v6-10 (0.26), v11-2 (0.19), v>20 (0.09)	v=0 (0.10), v1-5 (0.31), v6-10 (0.26), v11-2 (0.23), v>20 (0.09)
Scotland iii no	v=0 (0.13), v1-5 (0.54), v6-10 (0.24), v>10 (0.09)	v=0 (0.29), v1-5 (0.13), v6-10 (0.17), v11-2 (0.29), v>20 (0.11)	v=0 (0.24), v1-5 (0.26), v6-10 (0.21), v11-2 (0.19), v>20 (0.09)
Scotland iv (du	v=0 (0.22), v1-5 (0.44), v6-10 (0.25), v>10 (0.09)	v=0 (0.72), v1-5 (0.06), v6-10 (0.09), v11-2 (0.09), v>20 (0.03)	v=0 (0.31), v1-5 (0.16), v6-10 (0.19), v11-2 (0.22), v>20 (0.13)
Northern irelan	v=0 (0.11), v1-5 (0.49), v6-10 (0.33), v>10 (0.08)	v=0 (0.99), v11-2 (0.01)	v=0 (0.14), v1-5 (0.31), v6-10 (0.39), v11-2 (0.12), v>20 (0.04)

Bibliographie

[1] F.B. BAKER et L.J. HUBERT, *Measuring the power of hierarchical cluster analysis.* Journal of the American Statistical Association, 70, 31-38, 1975.

[2] E.M.L. BEALE, *Cluster analysis.* London : Scientific Control Systems, 1969.

[3] H.H. BOCK et E. DIDAY, *Analysis of Symbolic Data. Exploratory Methods for Extracting Statistical Information from Complex Data.* Springer-Verlag, 2000.

[4] T. CALINSKI et J. HARABASZ, *A dendrite method for cluster analysis.* Communications in Statistics, 3, 1-17, 1974.

[5] G. CELEUX, E. DIDAY, G. GOVAERT, Y. LECHEVALLIER et H. RALAN-BONDRAINY, *Classification automatique des données.* Dunod-Informatique, Bordas, 1989.

[6] J.L. CHANDON et S. PINSON, *Analyse typologique : théorie et applications.* Masson, Paris, 1981.

[7] S. DELOGNE, *Méthodes de détermination du nombre de classes pour des données symboliques de type intervalle.* FUNDP, 2002.

[8] R. O. DUDA et P. E. HART, *Pattern classification and scene analysis.* Wiley-Interscience, New York, 1973.

[9] B. EVERITT, *Cluster analysis.* Arnold, London, 1993.

[10] A. HARDY, *On the number of clusters.* Computational Statistics and Data Analysis, 23, 83-96, 1996.

[11] A. HARDY et P. LALLEMAND, *Determination of the number of clusters for symbolic objects described by interval variables.* Studies in Classification, Data Analysis and Knowledge Organisation, Springer-Verlag, 311-318, 2002.

[12] A. HARDY et J.P. RASSON, *Une nouvelle approche des problèmes de classification automatique.* Statistiques et Analyse des Données, 7, 41-56, 1982.

[13] L.J. HUBERT et J.R. LEVIN, *An examination of procedures for determining the number of clusters in a data set.* Psychometrika, 50(2), 159-179, Juin 1985.

[14] H.A.L. KIERS, J.P. RASSON, P.J.F. GROENEN, M. SCHADER, *Data Analysis, Classification, and Related Methods.* Springer-Verlag, 2000.

[15] S. SILVI et P.A. WAREMBOURG, *Datamining. Etude et analyse des ventes d'une chaîne de magasins.* Université Paris Dauphine, 2002. http ://www.ceremade.dauphine.fr/~touati/MAGASINwww/ RAPPORT_PROJET_E-FASHION.PDF